**MCA07**

# KRISHNA KANTA HANDIQUI STATE OPEN UNIVERSITY

**Housefed Complex, Dispur, Guwahati - 781 006**

## Master of Computer Applications

# FUNDAMENTALS OF DATABASE MANAGEMENT SYSTEM

## CONTENTS

## Subject Expert

Prof. Anjana Kakati Mahanta, Deptt. of Computer Science, Gauhati University
Prof. Jatindra Kr. Deka, Deptt. of Computer Science and Engineering,
Indian Institute of Technology, Guwahati
Prof. Diganta Goswami, Deptt. of Computer Science and Engineering,
Indian Institute of Technology, Guwahati

## Course Coordinator

Tapashi Kashyap Das, Assistant Professor, Computer Science, KKHSOU
Arabinda Saikia, Assistant Professor, Computer Science, KKHSOU

## SLM Preparation Team

| Units | Contributor |
|---|---|
| 1 | **Kshirod Sarma**,<br>Research Scholar, Rajiv Gandhi University,<br>Arunachal Pradesh, India |
| 2 | **Arabinda Saikia,** KKHSOU |
| 3, 4, 7 | **Pranab Das,**<br>Assistant Professor, Don Bosco University, Guwahati, Assam |
| 5, 6 | **Jonalee Barman Kakati** ,<br>Lecturer, Deptt. of Business Administration<br>NERIM, Guwahati, Assam |

***Dec. 2011***



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

The university acknowledges with thanks the financial support provided by the **Distance Education Council, New Delhi**, for the preparation of this study material.

**Housefed Complex, Dispur, Guwahati- 781006; Web: www.kkhsou.net**

## COURSE INTRODUCTION

This course is on "***Fundamentals of Database Management System***". This course is designed to give learners a clear understanding of database fundamentals providing all the major topics of the field. The course comprises of seven units which are as follows:

**Unit - 1** introduces file structure and organization. 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** is on database management system. This unit gives you the idea of DBMS, its advantages and disadvantages, DBMS users, DBMS language etc.

**Unit- 3** concentrates on data models. Data Models like Object based models, Relational models, Network models, Hierarchical models etc. are discussed in this unit.

**Unit - 4** is on relational database. Various concepts associated with relational database like tuples, attributes, cardinality, degree etc. along with the concept of keys and relational algebra are discussed in this unit.

**Unit - 5** and **Unit - 6** are on structured query language. With these units learners will be able to write SQL systax using various SQL commands.

**Unit - 7** deals with relational database design. Concepts of normalization and various normal forms are described in this unit

Each unit of this course 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 additional interesting and relevant information. Again, you will get "CHECK YOUR PROGRESS" questions. These have been designed to make you 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.

# MASTER OF COMPUTER APPLICATIONS

## Fundamentals of Database Management System

## DETAILED SYLLABUS

**Unit 1: File Structure and Organization** (Marks: 12)

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 Management System** (Marks: 15 )

Definition of DBMS, File processing system vs DBMS, Advantages and Disadvantages of DBMS, Database Architecture, Data Independence, Data Dictionary, DBMS Language, Database Administrator

**Unit 3: Data Models** (Marks: 15)

Data Models: Object Based Logical Model, Record Base Logical Model, Relational Model, Network Model, Hierarchical Model, Entity-Relationship Model : Entity Set, Attribute, Relationship Set, Entity Relationship Diagram (ERD), Extended features of ERD

**Unit 4 : Relational Databases** (Marks: 15 )

Relational data model; Terms :Relation, Tuple, Attribute, Cardinality, Degree, Domain; Keys : Super Key, Candidate Key, Primary Key, Foreign Key; Relational Algebra- Operations: Select, Project, Union, Difference, Intersection, Cartesian Product, Natural join

**Unit 5 SQL (Part I)** (Marks: 14)

Introduction of SQL, characteristis of SQL, Basic Structure, DDL Commands, DML, DQL, SELECT Statement, WHERE Clause, Useful Relational Operators, Aggregate Functions, SUM Function, AVG Function

**Unit 6: SQL (Part II)** (Marks: 14)

Compound Conditions and Logical Operators, AND Operator, OR Operator, Combining AND and OR Operators, IN Operator, BETWEEN Operator, NOT Operator, Order of Precedence for Logical Operators, LIKE Operator, Concatenation Operator, Alias Column Names, ORDER BY Clause, Handling NULL Values, DISTINCT Clause

**Unit 7: Relational Database Design** (Marks: 15)

Introduction to Normalization, Anomalies of unnormalized database, Normal Form : 1NF, 2NF, 3 NF

*****

# UNIT 1 : FILE STRUCTURE AND ORGANIZATION

## UNIT STRUCTURE



## 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

## 1.2 INTRODUCTION

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

approach for data processing is to store locally needed data and develop program for each type of application. In the traditional file processing approach data for multiple applications may not be integrated and manipulated in a single file. For such reasons organizations are migrating from traditional manual system to a computerised information system for which the data within the organization is a basic resource. For the formation of computerised record keeping system we are required to introduce database and database management systems (DBMS). Some basic concepts about 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 infomation are closely related and are often used interchangeably. *Data* may be defined as a known fact that can be recorded and that have implicit meaning. It can be anything like name of a person or a place of stay or a number representing roll number of student, ticket number of passenger, passport number of an immigrant, bill number for a payee, credit card number for a holder, identification number for an employee etc. Data is the name given to basic facts and entities such as names and numbers. There are uncountable examples of data such as weights, prices, costs, numbers of items sold or purchased, employee names, product names, addresses, tax codes, registration marks etc.


**Fig. 1.1**

*Information* is data that has been converted or processed into a more useful form. It is the set of data that has been organized for direct utilization of mankind, as information helps human beings in their decision making process. Examples are Time Table, Merit List, Report card, Headed



DATA PROCESSING INFORMATION

tables, printed documents, pay slips, receipts, reports etc. The information is obtained by assembling items of data into a meaningful forms. For example, marks obtained by students and their roll numbers form *data*, the report card/sheet is the *information*. Other forms of information are pay-slips, schedules, reports, worksheet, bar charts, invoices and accounts returns etc. It may be noted that information containing wisdom is known as *knowledge* or it is applied form of information. Information becomes utility oriented only when it is applied in particular area of expertise or specialization.

## 1.4 FIELDS AND RECORDS

The smallest piece of meaningful information is called a ***field*** or ***data item***. In a telephone bill, Name, Telephone_no, Bill_amount, Address are few examples of fields.

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. A record consists of values for each field. Records can be classified according to their length. It may be ***fixed length record*** or ***variable length record***.

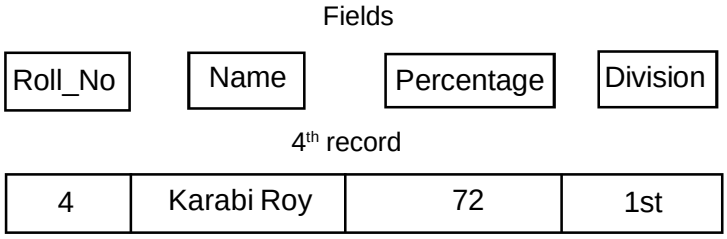
Each data item or field in a fixed length record is given a fixed length. The length of the record should be large enough to hold even the largest value anticipated for that field. In a student's record we may provide 15 spaces for the name field, 3 for roll number, 2 for each percentage field, 4 for 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 of a fixed size. The value in every record is allowed to take up as much space as is required by its size. 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 (19 characters)

## 1.5 FILES

A database ***file*** is a collection of related sequence of records. In many cases, all records in a file are of the same record type i.e., (records with identical format). If every record in a file has exacly the same size in bytes then the file is said to be maid up of *fixed-length records*. If different records in the file have different sizes, the file is said to be made of *variable-length records*. A stock file will contain all the records of every item available in stock.




File

| Roll No. | Name | Percentage | Division |
|---|---|---|---|
| 1 | Gautam Baruah | 72 | 1st |
| 2 | Pritam Kashyap | 68 | 1st |
| 3 | Rajib Sharma | 44 | 3rd |
| 4 | Karabi Roy | 72 | 1st |
| 5 | Debojit Saikia | 81 | 1st |
| 6 | Bhavna Das | 60 | 1st |
| 7 | Sweta Mishra | 55 | 2nd |
| 8 | Manoj Bora | 76 | 1st |
| 9 | Pritam Kashyap | 83 | 1st |
| 10 | Mriganka Bharali | 42 | 3rd |

Field content

**Fig. 1.2 : Concept of field, records and files**

For example, in the following figure 1.2, a srudent 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.*** The collection of informations 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. 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 fig(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 following 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, address of a student would appear on a student master file.

A transaction file is a file in which the current data are stored for subsequent processing usually in combination with a master file. Transaction file contains the records that are used to update the records of the master file. For example, a transaction file may contain the roll number of students whose home address has recently changed. This file will used to 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. It is very important that the transaction file should contain records with the fields and their names in the same order and size and type of the master file. Otherwise the process of updating of the master file will cause an error.

### 1.5.1 Operation on Files

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 student can be treated as a search key.

A computer can also work in the same way. Instead of paper files, computer will store electronic files in a hard disk or removable disk. 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. There are various operations associated with computerised files. The major operations on files are given below :

**Updating of file :** Updating is the process of modifying a file with current information according to a specified procedure. The update operation may include:

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.

*ne in*
*s are*

**File maintenance :** New records must be altered in a file. For example, student's adresses also change and new addresses have to be inserted to bring the file up to date. These particular activities come under the heading of 'maintaining' the file. File maintenance can be carried out as a separate run, but the insertion and deletion of records are sometimes combined with updating.

**File enquiry :** This is the operation of both master and transaction files to obtain information contained therein. It involves the need to acertain a peice of information from, say, a master record.This does not involve any alteration to the file contents.

## 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 sequence order. For example, a student's record file may be kept in order of ascending 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 any where 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 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. The main drawbacks of this type of organization are:

- Sequential file are not suitated 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 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.

  Some advantages of sequential file organization are:
- 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

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

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:

**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.

### 1.6.4 Heap Files

Basically heap files are unordered files. It is the simplest and most basic type. These files consist of randomly ordered records. The records will have no particular order. The operations we can perform on the records are insert, retrieve and delete. The features of the heap file are:

- New records can be inserted in any empty space that can accommodate them.
- When old records are deleted, the occupied space becomes empty and available for any new insertion.
- If updated records grow; they may need to be relocated (moved) to a new empty space. This needs to keep a list of empty space.

**Advantages of heap files**

1. This is a simple file Organization method.
2. Insertion is somehow efficient.
3. Good for bulk-loading data into a table.
4. Best if file scans are common or insertions are frequent.

**Disadvantages of heap files**

1. Retrieval requires a linear search and is inefficient.
2. Deletion can result in unused space/need for reorganization.

### 1.6.5 Hash Files

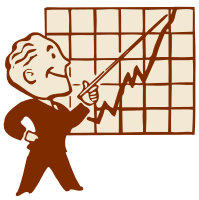
Hashing is the most common form of purely random access to a file or database. It is also used to access columns that do not have an index as an optimization technique. Hash functions calculate the address of the page in which the record is to be stored based on one or more fields in the record. The records in a hash file appear randomly distributed across the available space. It requires some hashing algorithm and the technique. Hashing Algorithm converts a primary key value into a record address. The most popular form of hashing is division hashing with chained overflow.

**Advantages of Hashed file Organisation**

1. Insertion or search on hash-key is fast.
2. Best if equality search is needed on hash-key.

**Disadvantages of Hashed file Organization**

1. It is a complex file Organization method.
2. Search is slow.
3. It suffers from disk space overhead.
4. Unbalanced buckets degrade performance.
5. Range search is slow.



**CHECK YOUR PROGRESS**

Q.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 constitute a file.

iv) A file needed for updating a master files 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 file 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: relative and absolute 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.
- Basically **heap** files are unordered files. It is the simplest and most basic type. These files consist of randomly ordered records.
- **Hashing** is the most common form of purely random access to a file or database. It is also used to access columns that do not have an index as an optimization technique.

## 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 READINGS


- *Database Systems:Concepts, Design and Applications,* by S.K.Singth, Pearson Education
- *Introduction to Database Management Systems*, by Atul Kahate, Pearson Education
- *Fundamentals of Database Systems*, by Elmasri Navathe, Somayajulu & Gupta , Pearson Education Publication.
- *An Introduction to Database Systems*, by C.J. Date, Eighth Edition, Addison Wesley, 2003.

## 1.10 MODEL QUESTIONS

Q.1. Explain data and information?

Q.2. What do you mean by the terms: field, record, file. Explain with examples.

Q.3. What are the different types of file organization techniques?

Q.4. Write short notes on :

a) Direct access organization

b) Sequential access organization

c) Index-Sequential organization

d) Hash file organization

Q.5. What are the advantages and disadvantages of sequential access organization.

Q.6. What are the advantages and disadvantages of sequential access organization.

Q.7. What are advantages and disadvantages of heap files?

# UNIT 2 : DATABASE MANAGEMENT SYSTEM

## UNIT STRUCTURE



## 2.1 LEARNING OBJECTIVES

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

- define database and DBMS
- describe DBMS Architecture
- illustrate data independence and data dictionary
- explain DBMS language
- describe the role of DBA

## 2.2 INTRODUCTION

In the previous unit, we have learnt about the basic idea of data, information, fields and records. In addition, concepts of files and basic file organization techniques are also introduced. All these are the elementary concepts relating to the database system.

In this unit, we will learn the concept of *database* and *DBMS*. We will also discuss the three-tier database architecture and different types of DBMS languages. Moreover, responsibilitties of database administrator will also be discussed at the end of this unit.

## 2.3 FILE BASED APPROACH

In case of traditional file based approach, an organization's information was stored as group of records in separate files. These file processing systems consist of data files and many application programs. Each file, termed as a *flat file,* contains and processes information for one specific function, such as accounting or inventory. Programming languages such as COBOL were used by the programmers to write application programs. Each application files and programs were created and maintained independent of another 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 format in the above applications
- whenever some information regarding a group of students are 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 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.

In the following figure (Fig.2.1), a traditional record keeping system of a University is shown, where every applications are interrelated and caused repetation of same data in different files which leads to the problem of data redundancy and inconsistency.

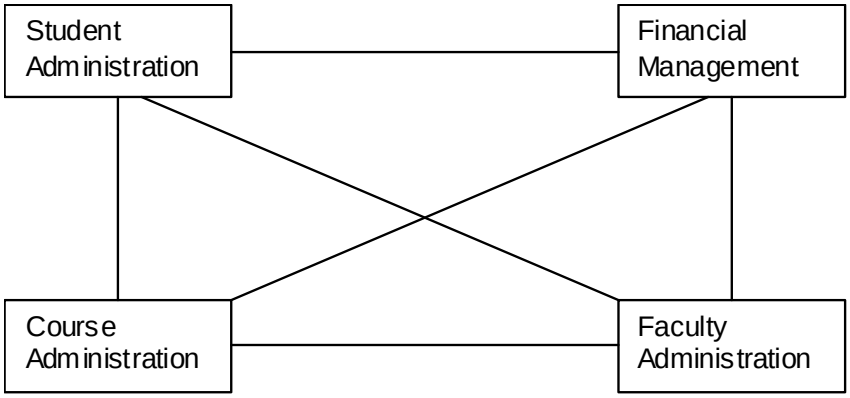



Fig. 2.1 Traditional File Approach

Some of the disadvantages of file based approach are discussed below:

**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 same data in more than one file is known as data redundancy. This leads to increase in cost of data entry and data storage.

**Data integrity problem :** Data integrity refers to 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 were described by specific physical formats that were coded into the application program by programmer. If the format of a certain record was 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.

## 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 database approach the user is free from the detailed and complicated task of keeping up with the physical structure of data. A database approach is shown by the following diagram(Fig.2.2).

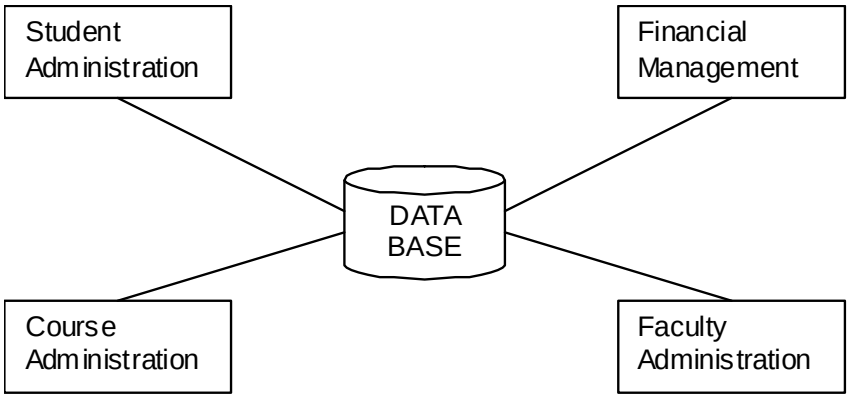



Fig. 2.2 : Database approach

Here, it is shown that several applications share common data in a database approach.

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, it

- is a collection of interrelated data stored together without harmflul or unnecessary 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.

### 2.4.1 Advantage of Database Approach

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 in 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 to each other. However, the database approach provides an independence between the file structure and program structure.

- **Sharing of data and Data security :** Data in a database are shared among users and applications. In database approach data are protected from unauthorised access by some security mechanism.

## 2.5 DATABASE MANAGEMENT SYSTEM

The ***database management system****(DBMS)* is the interface between the users(application programmers) and the database(the data). A database management system is a program that allows 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 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*.

### 2.5.1 Components of DBMS

The functional components of DBMS are :

- *Storage manager*
- *Query processor*

***Storage manager*** is responsible for storing, retrieving and updating data in database. Storage manager components are :

*Authorization and integrity manager* : Checks the integrity constraints and authority of users to access data.

*Transaction manager* : Transaction means a collection of operations that performs a single logical function in a database application. It ensures that the database remains in a consistent state if there is a transaction failure or system failure.

*File manage*r : It is responsible for allocating space on disk storage. The files are used for storing collections of similar data. The file manager can create and delete file also update and retrieve records from files.

*Buffer manager :* The buffer is a temporary memory space from which a file is read. The buffer manager is responsible for fetching data from disk storage into main memory.

**Query processor :** Query processing is a procedure of converting a query written in high level language (eg. SQL, QBE) into a correct and efficient execution plan expressed in low level language, which is used for data manipulation. The success of a query language also depends upon its query processor.

### 2.5.2 Advantages of DBMS

Due to the centralised management and control, the database management system has numerous advantages, some

of which are explained below :

**Minimal data redundancy :** Centralized control of data avoids unnecessary duplication of 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.

**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.

**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.

**Improved data sharing :** Since, database system is a centralised repository of 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.

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

**Data security :** Database security is the protection of database from unauthorised users. The DBA(DataBase Administrator, we will discuss the responsibility of DBA in the next section) can define security rules to chueck unauthorised access to data. Some users may be given rights to only retrieve data, 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.

**Enforcement of standards :** With the central control of the database, a DBA can defines and enforces 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 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.

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

### 2.5.3 Disadvantages of DBMS

Some of the disadvantages of DBMS are as follows:

**Complexity :** The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software. Database designers, developers, database administrators and end-users must understand this functionality to take full

advantage of it. Failure to understand the system can lead to bad design decisions, which can have serous consequences for an organization.

**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.

**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 many may not run as fast as they used to.

**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.

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

**Additional Hardware cost :** The disk storage requirements for the DBMS and the database many 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.

**Cost of Conversion :** The cost of the DBMS and extra hardware may be significant compared with 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**

Q.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

Q.2.What is database? Give example.

Q.3.Define DBMS.

## 2.6 DATABASE ARCHITECTURE

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 users with an abstract view of the data. That is the system hides certain details of how the data is stored and maintained. 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 following figure(Fig.2.3.)

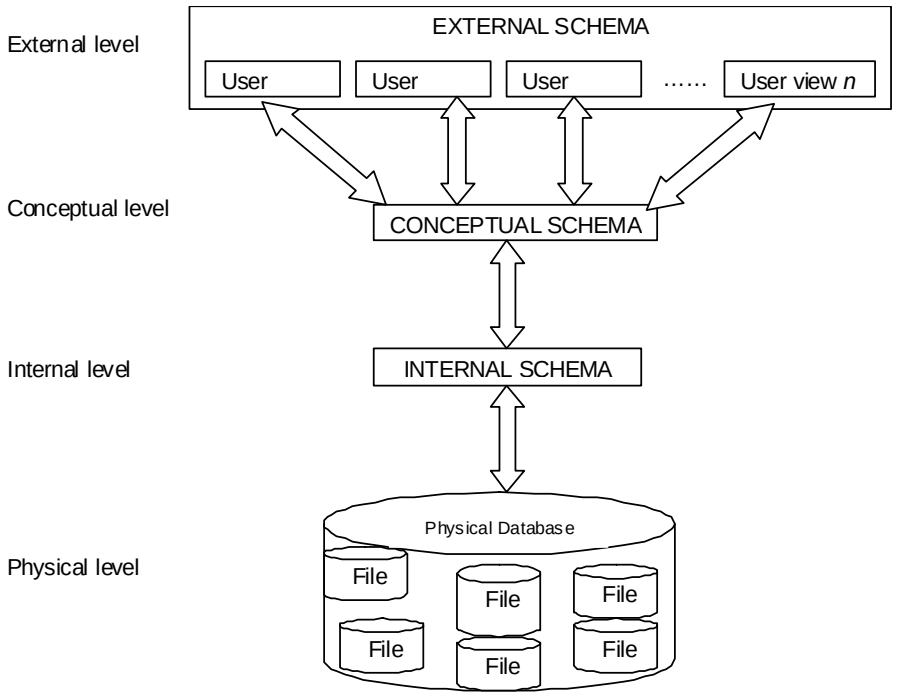



Fig. 2.3 : Three-tier database architecture

We can imagine that the whole database system is divided into levels. These are:

- 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 describes the different views of the database (for customer) ifferen (for purchase manager)

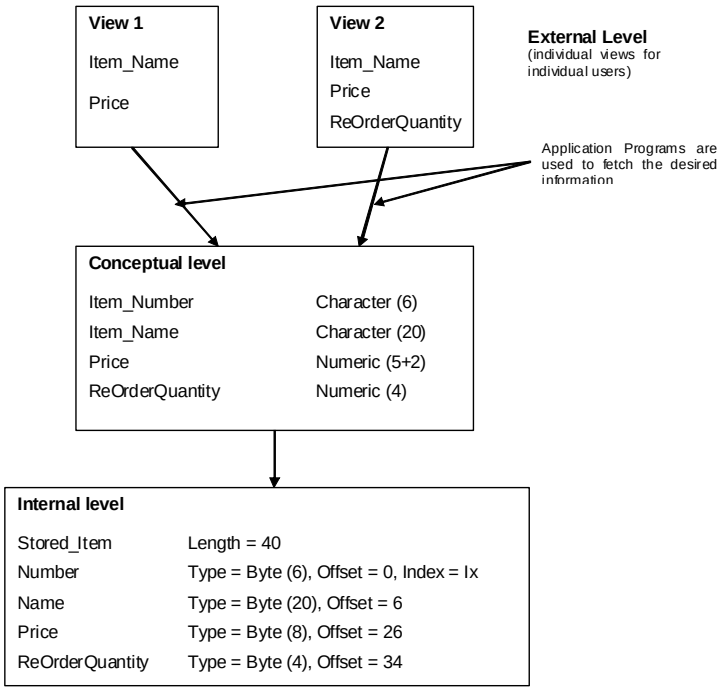



Fig. 2.4 : View of data at three-tier database architecture

**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 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 describe 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.

## 2.7 DATA INDEPENDENCE

***Data independence*** is the characteristics 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 -

- Logical data independence
- Physical 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 refers to the external schema construct must work as before.

***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.

**NOTE**

Practically, DDL & DML are not two separate languages, instead they simply form parts of a single database language. SQL represents combination of DDL, DML and VDL.

## 2.8 DATA DICTIONARY

A *data dictionary* also known as a *system catalog i*s a centralized store of information about the database. It contains the information about the tables, the fields and the table contain – the data types, primary keys, indexes, the joins which have been established between those tables, referential integrity, cascades update, cascade delete etc. We will come across with these terms in a later unit. The information stored in the data dictionary is called the ***metadata***. Thus, a data dictionary can be considered as a file that stores metadata. Data dictionary is a tool for recording and processing information about the data that an organization uses. The data dictionary is a central catalog for metadata. The data dictionary can be integrated within the DBMS or separate.

## 2.9 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 database, a DBMS is choosen to implement the database. It is important to first specify the conceptual and internal schemas for the database. Following laguages are used for specifying database schemas :

- Data definition language (DDL)
- Storage definition language (SDL)
- View definition language (VDL)
- Data manipulation language (DML)

**Data definition language (DDL) :** DDL is a special language which specify the database conceptual schema using set of definitions. DDL allows the DBA or 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 inorder 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 a EMPLOYEE table as shown below :

| EMPLOYEE | | | | |
|---|---|---|---|---|
| Fname | Lastname | Eno | DOB | Address |
| | | | | |
| | | | | |
| | | | | |

**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.

**View definition language (VDL) :** View definition language is used to specify user's views(external s chema) 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).

**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 applies 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.

## 2.10 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 batabase system who designs database, controls and manages all the resources of database as well as provides necessary technical support for implementing lolicy 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 most 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

Q.4. 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.

Q.5. 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

Q.6. What is meant by metadata ?

Q.7. Define the term data dictionary.

Q.8. what is meant by physical and logical data independnce?

Q.9. Define the concept of database schema? Write the names of the schemas that exists in a database complying with the three levels of ANSI/SPARC architecture.

## 2.11 LET US SUM UP

- 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.
- 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.
- A database management system (DBMS) is a collection of programs that enables users to store, modify and extract information from a database as per the requirements. DBMS is an intermediate layer between programs and the data. Programs access the DBMS, which then accesses the data.
- 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
- Logical data independence indicates that the conceptual schema

can be changed without affecting the existing external schema.

- 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.
- A data dictionary also known as a system catalog is a centralized store of information about the database.
- DBMS provide appropriate languages and interfaces for each category of users to express database queries and updates.
- Following laguages are used for specifying database schemas :
  - Data definition language (DDL)
  - Storage definition language (SDL)
  - View definition language (VDL)
  - Data manipulation language (DML)
- A DBA provides the necessary technical support for implementing policy decisions of database.

## 2.13 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1 :** a. (iv), b. (iv), c. (ii), d. (i), e. (i)

**Ans. to Q. No. 2 :** A database is a collection of related data. Here, the term 'data' means that known facts that can be record. Examples of database are library information system, railway, bus or airline reservation system etc.

**Ans. to Q. No. 3 :** DBMS is a collection of programs that enables users to create and maintain a database.

**Ans. to Q. No. 4 :** (i) F, (ii) T, (iii) F, (iv) F, (v) T, (vi) T

**Ans. to Q. No. 5 :** a. (ii), b. (i), c. (ii), d. (ii), e. (i)

**Ans. to Q. No. 6 :** Metadata are data about the data but not the actual data.

**Ans. to Q. No. 7 :** Data dictionary is a file that contains metadata.

**Ans. to Q. No. 8 :** In logical data independence, the conceptual schema can be changed without changing the external schema. In physical data independence, the internal schema can be changed without changing the conceptual schema.

**Ans. to Q. No. 9 :** Database schema is nothing but description of the database. The types of schemas that exist in a database complying with three levels of ANSI/SPARC architecture are : *external schema, conceptual schema and internal schema.*

## 2.14 FURTHER READINGS

- *An Introduction to Database Systems*, C. J. Date, Pearson Education.
- *An Introduction to Database Systems*, B.C. Desai.
- *Database System Concepts*, S. K. Singh, Pearson Education.
- *Principles of Database systems*, J.D. Ullman.

## 2.15 MODEL QUESTIONS

Q.1. What is file based approach of database? Explain its limitations?

Q.2. Explain three level database architecture. What are its objectives?

Q.3. What do data independence and its types? How data indep-endence is achieved?

Q.4. What are advantages of DBMS?

Q.5. Discuss the main disadvantages of a Traditional file approach?

Q.6. Discuss the main disadvantages of DBMS?

Q.7. Difference between DBMS approach & traditional file approach.

Q.8. Mention the differences between text files and database files. Why

are database files preferred in a commercial organization?

Q.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

Q.10. What is logical data independence and why is it important?

Q.11. Explain the difference between logical and physical data independence.

Q.12. Describe the three levels of data abstraction?

Q.13. What do you mean Database Language? What are the different types of data base language?

Q.14. Explain the role of a Database Administrator.

# UNIT 3 : DATA MODELS

## UNIT STRUCTURE



## 3.1 LEARNING OBJECTIVES

After going through this unit, you will able to

- define data model
- learn about the importance of data model
- the most common types of data models
- learn about Entity-Relationship model
- learn how to draw ER diagram
- learn about extended ER

## 3.2 INTRODUCTION

To begin our discussion of data models we should first begin with a common understanding of what exactly we mean when we use the term. A data model is a picture or description which depicts how data is to be arranged to serve a specific purpose. The data model depicts what that

data items are required, and how that data must look. However it would be misleading to discuss data models as if there were only one kind of data model, and equally misleading to discuss them as if they were used for only one purpose. It would also be misleading to assume that data models were only used in the construction of data files.

Some data models are schematics which depict the manner in which data records are connected or related within a file structure. These are called record or structural data models. Some data models are used to identify the subjects of corporate data processing - these are called entity-relationship data models. Still another type of data model is used for analytic purposes to help the analyst to solidify the semantics associated with critical corporate or business concepts.

Although the term data modeling has become popular only in recent years, in fact modeling of data has been going on for quite a long time. It is difficult for any of us to pinpoint exactly when the first data model was constructed because each of us has a different idea of what a data model is. If we go back to the definition we set forth earlier, then we can say that perhaps the earliest form of data modeling was practiced by the first persons who created paper forms for collecting large amounts of similar data. We can see current versions of these forms everywhere we look. Every time we fill out an application, buy something, make a request on using anything other than a blank piece of paper or stationary, we are using a form of data model.These forms were designed to collect specific kinds of information, in specific format.

## 3.3 DATA MODEL

A model is a representation of reality, 'real world' objects and events, and their associations. It is an abstraction that concentrates on the essential, inherent aspects of an organization and ignore the accidental properties.

A data model is a collection of high-level data description constructs that hide many low-level storage details, it also describe data, its relationships, and its constraints and provides a clearer and more accurate

description and representation of data. A DBMS allows a user to define the data to be stored in terms of a data model. Most database management systems today are based on the relational data model.

A data model comprises of three components:

- A structural part, consisting of a set of rules according to which databases can be constructed.
- A manipulative part. Defining the types of operation that are allowed on the data ,this includes the operations that are used or updating or retrieving data from the database and for changing the structure of the database.
- Possibly a set of integrity rules, which ensures that the data is accurate.

Data models vary in both complexity and richness. However, all data models are equivalent as far as their ability to model information is concerned. What is more important as far as selecting a model is concerned is matching the inherent structure of the problem being modeled. This structure varies as the problem is investigated and refined. In the beginning, when little is known about what the final model will be, the simplest, most flexible and least structured scheme provides the greatest freedom of expression. As time passes, it becomes more important to fashion the model using a scheme that closely matches the final implementation.

It's important to remember that data models are used for both conceptual and implementation purposes. Emphasizing one over the other may distort how one perceives the way models fit together. No one size fits all situations. Each model has strengths and weaknesses.

## 3.4 NEED FOR DATA MODEL

The purpose of a data model is to represent data and to make the data understandable.

- Data models represents complex real-world data structures
- Facilitate interaction among the designer, the applications programmer and the end users
- End-users have different views and needs for data

- Data model organizes data for various users

## 3.5 TYPES OF DATA MODEL

There have been many data models proposed in the literature. They fall into three broad categories:

- Record Based Data Models
- Object Based Data Models
- Physical Data Models

**Record Based Data Models**: A record based data model is used to specify the overall logical structure of the database. In this model the database consists of a no. of fixed formats of different types.Each record type defines a fixed no. of fields having a fixed length.

There are 3 principle types of record based data model. They are:

1. Hierarchical data model.
2. Network data model.
3. Relational data model.

**Hierarchical Model :** The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments. This structure implies that a record can have repeating information, generally in the child data segments. Data in a series of records, which have a set of field values attached to it. It collects all the instances of a specific record together as a record type. These record types are the equivalent of tables in the relational model, and with the individual records being the equivalent of rows. To create links between these record types, the hierarchical model uses *Parent Child* Relationships. These are a 1:N mapping between record types. This is done by using trees, like set theory used in the relational model, "borrowed" from maths.

For example, an organization might store information about an employee, such as name, employee number, department, salary. The organization might also store information about an employee's children, such as name and date of birth. The employee and children data forms a hierarchy, where the employee data represents the parent segment and the children data represents the child segment. If an employee has three

children, then there would be three child segments associated with one employee segment. In a hierarchical database the parent-child relationship is one to many. This restricts a child segment to having only one parent

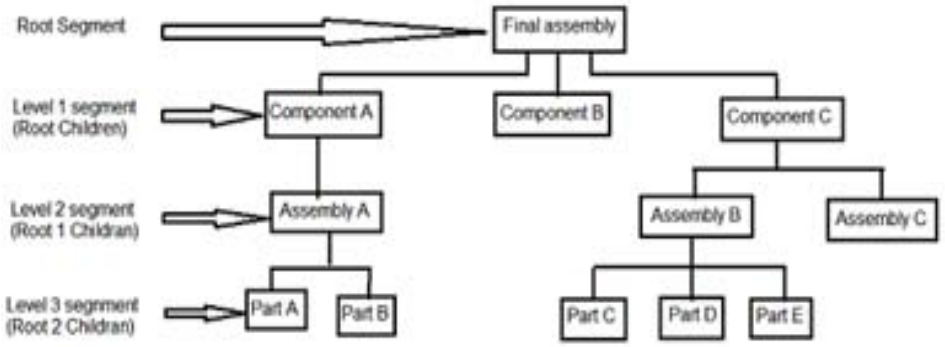



**Fig. 3.1 : A Hierachical Structure**

Hierarchical DBMSs were popular from the late 1960s, with the introduction of IBM's Information Management System (IMS) DBMS, through the 1970s.

**Advantages**

- Many features form the foundation for current data models
- Generated a large installed base of programmers
- Who developed solid business applications

**Disadvantages**

- Complex to implement
- Difficult to manage
- Lacks structural independence
- Implementation limitations
- Lack of standards (Company vs. Industry or Open)

**Network Model :** The popularity of the network data model coincided with the popularity of the hierarchical data model. Some data were more naturally modeled with more than one parent per child. So, the network model permitted the modeling of many-to-many relationships in data. In 1971, the Conference on Data Systems Languages (CODASYL) formally defined the network model. The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type. A member record type can have that role in more

than one set, hence the multiparent concept is supported. An owner record type can also be a member or owner in another set. The data model is a simple network, and link and intersection record types may exist, as well as sets between them . Thus, the complete network of relationships is represented by several pairwise sets; in each set some (one) record type is owner (at the tail of the network arrow) and one or more record types are members (at the head of the relationship arrow). Usually, a set defines a 1:M relationship, although 1:1 is permitted. The CODASYL network model is based on mathematical set theory.

**Advantages**

- The network model is conceptually simple and easy to design.
- Ability to handle more relationship types
- The network model can handle the one-to-many and many-to-many relationships.
- Ease of data access in the network database terminology, a relationship is a set. Each set comprises of two types of records:- an owner record and a member record, In a network model an application can access an owner record and all the member records within a set.
- Data integrity in a network model, no member can exist without an owner. A user must therefore first define the owner record and then the member record. This ensures the integrity.
- Data Independence - The network model draws a clear line of demarcation between programs and the complex physical storage details. The application programs work independently of the data. Any changes made in the data characteristics do not affect the application program.

**Disadvantage**

- System complexity – In a network model, data are accessed one record at a time.It is essential for the database designers, administrators, and programmers to be familiar with the internal data structures to gain access to the data.Therefore, a user friendly database management system cannot be created using

the network model

- Lack of Structural independence – Making structural modifications to the database is very difficult in the network database model as the data access method is navigational. Any changes made to the database structure require the application programs to be modified before they can access data. Though the network model achieves data independence, it still fails to achieve structural independence.

**Relational Model :** The Relational Model is a clean and simple model that uses the concept of a relation using a table rather then a graph or shapes. The information is put into a grid like structure that consists of columns running up and down and rows that run from left to right, this is where information can be categorized and sorted.

Properties of Relational Tables:

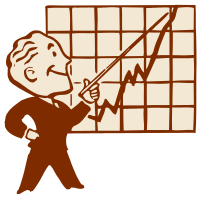
1. Values Are Atomic
2. Each Row is Unique
3. Column Values Are of the Same Kind
4. The Sequence of Columns is Insignificant
5. The Sequence of Rows is Insignificant
6. Each Column Has a Unique Name

Certain fields may be designated as keys, which means that searches for specific values of that field will use indexing to speed them up. Where fields in two different tables take values from the same set, a join operation can be performed to select related records in the two tables by matching values in those fields. Often, but not always, the fields will have the same name in both tables.

For example, an "orders" table might contain (customer-ID, product-code) pairs and a "products" table might contain (product-code, price) pairs so to calculate a given customer's bill you would sum the prices of all products ordered by that customer by joining on the product-code fields of the two tables. This can be extended to joining multiple tables on multiple fields.

**Physical Data Model :** A physical data model is a representation of a data design which takes into account the facilities and constraints of a

given database management system. In the lifecycle of a project it is typically derived from a logical data model, though it may be reverse-engineered from a given database implementation. A complete physical data model will include all the database artifacts required to create relationships between tables or achieve performance goals, such as indexes, constraint definitions, linking tables, partitioned tables or clusters. The physical data model can usually be used to calculate storage estimates and may include specific storage allocation details for a given database system.



## CHECK YOUR PROGRESS

Q.1. Select the correct answer :

i) A top-to-bottom relationship among the items in a database is established by a

A) hierarchical schema B) network schema

C) relational schema D) all of the above

ii) The highest level in the hierarchy of data organization is called

A) data bank B) data base

C) data file D) data record

iii) The relational database environment has all of the following components except

A) users B) separate files

C) database D) query languages

iv) One approach to standardization storing of data?

A) MIS B) structured programming

C) CODASYL specification D) none of the above

v) A collection of concepts that can be used to describe the structure of a database is called a

A) Database B) DBMS

C) Data model D) Data

vi) SQL was developed as an integral part of

A) A hierarchical database B) A data warehouse

C) A relational database D) All of them

## 3.6 ENTITY- RELATIONSHIP MODEL

The ***Entity-Relationship*** (ER) model allows us to describe the data involved in a real-world enterprise in terms of objects and their relationships and is widely used to develop an initial database design.The ER model is important primarily for its role in database design. It provides useful concepts that allow us to move from an informal description of what users want from their database to a more detailed, and precise, description that can be implemented in a DBMS.

### 3.6.1 Entities, Entity Sets and Attribute

An ***entity*** is an object that exists and is distinguishable from other objects. For instance, Ram Mohan with S.I.N. 890-12-3456 is an entity, as he can be uniquely identified as one particular person in the universe. An entity may be concrete (a person or a book, for example) or abstract (like a holiday or a concept).

An ***entity set*** is a set of entities of the same type (e.g., all persons having an account at a bank). Entity sets need not be disjoint. For example, the entity set employee (all employees of a bank) and the entity set customer (all customers of the bank) may have members in common.

An entity is represented by a set of ***attributes***. For example, name, S.I.N., street, city for ``customer'' entity.

The ***domain*** of the attribute is the set of permitted values (e.g., the telephone number must be seven positive integers).

Formally, an *attribute* is a function which maps an entity set into a domain.Every entity is described by a set of (attribute, data value) pairs.There

is one pair for each attribute of the entity set.

E.g. a particular customer entity is described by the set {(name, Harris), (S.I.N., 890-123-456), (street, North), (city, Georgetown)}.

**Relationships & Relationship Sets :** A relationship is an association between several entities.A relationship set is a set of relationships of the same type.

Formally it is a mathematical relation on n>1(possibly non-distinct) sets.

If $E_1$,$E_2$,.....$E_n$ are entity sets, then a relationship set R is a subset of {($e_1$,$e_2$,.....$e_n$)| $e_1 \varepsilon E_1$.....$e_n \varepsilon E_n$ } where, ($e_1$,$e_2$,.....$e_n$)is a relationship.

For example, consider the two entity sets customer and account. We define the relationship CustAcct to denote the association between customers and their accounts. This is a binary relationship set.

Going back to our formal definition, the relationship set CustAcct is a subset of all the possible customer and account pairings.This is a binary relationship. Occasionally there are relationships involving more than two entity sets.

The role of an entity is the function it plays in a relationship. For example, the relationship works-for could be ordered pairs of employee entities. The first employee takes the role of manager, and the second one will take the role of worker.

A relationship may also have descriptive attributes. For example, date (last date of account access) could be an attribute of the CustAcct relationship set

**Entity Relationship Diagrams :** An entity-relationship diagram is a data modeling technique that creates a graphical representation of the entities, and the relationships between entities, within an information system.Entity Relationship Diagrams (ERDs) illustrate the logical structure of databases.

There are three basic elements in ER models:

**Entities** are the "things" about which we seek information.An entity may be a physical object such as a house or a car, an event such as a house

sale or a car service, or a concept such as a customer transaction or order.

**Attributes** are the data we collect about the entities.

**Relationships** provide the structure needed to draw information from multiple entities.A relationship captures how two or more entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: an owns relationship between a company and a computer, a supervises relationship between an employee and a department.

Every entity must have a minimal set of uniquely identifying attributes, which is called the entity's primary key.Entities and relationships can both have attributes. Examples: an employee entity might have a Social Security Number (SSN) attribute; the proved relationship may have a date attribute.

Entity-relationship diagrams don't show single entities or single instances of relations. Rather, they show entity sets and relationship sets. Example: a particular song is an entity. The collection of all songs in a database is an entity set. The eaten relationship between a child and her lunch is a single relationship. The set of all such child-lunch relationships in a database is a relationship set. In other words, a relationship set corresponds to a relation in mathematics, while a relationship corresponds to a member of the relation.

The cardinality defines the relationship between the entities in terms of numbers. An entity may be optional: for example, a sales rep could have no customers or could have one or many customers.The three main cardinal relationships are: one-to-one, expressed as 1:1; one-to-many, expressed as 1:M; and many-to-many, expressed as M:N.

The steps involved in creating an ERD are:

- Identify the entities.
- Determine all significant interactions.
- Analyze the nature of the interactions.
- Draw the ERD.

A number of CASE tools, such as Visible Analyst and Data Architect,

can be used to generate ERDs.

## 3.7 ER MODEL

**ER Diagram Symbols :**

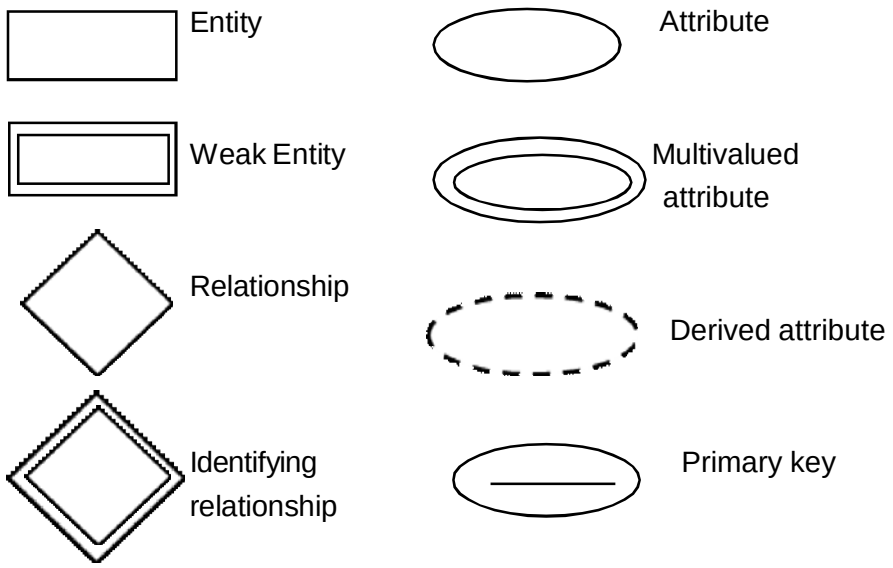



**Fig. 3.2**

**Simple versus composite :** Simple attributes are atomic.E.g. tel_ph; house color; basic-salary.Composite attributes made up of simple attributes.

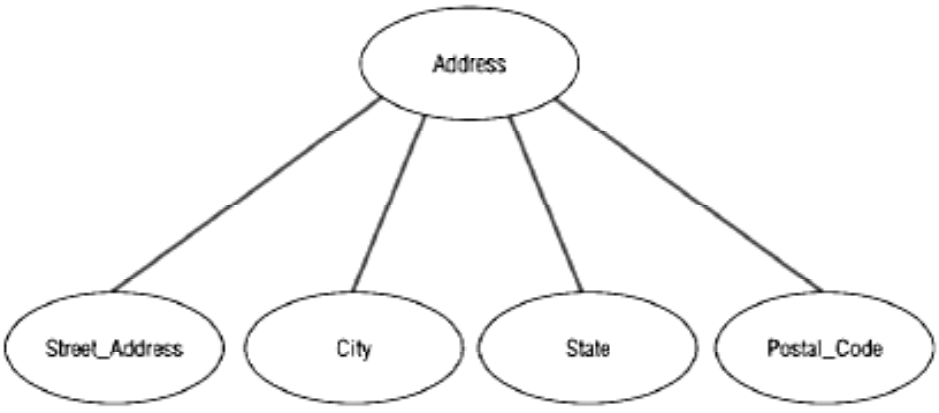



**Fig. 3.3**

**Simple Attribute example :**

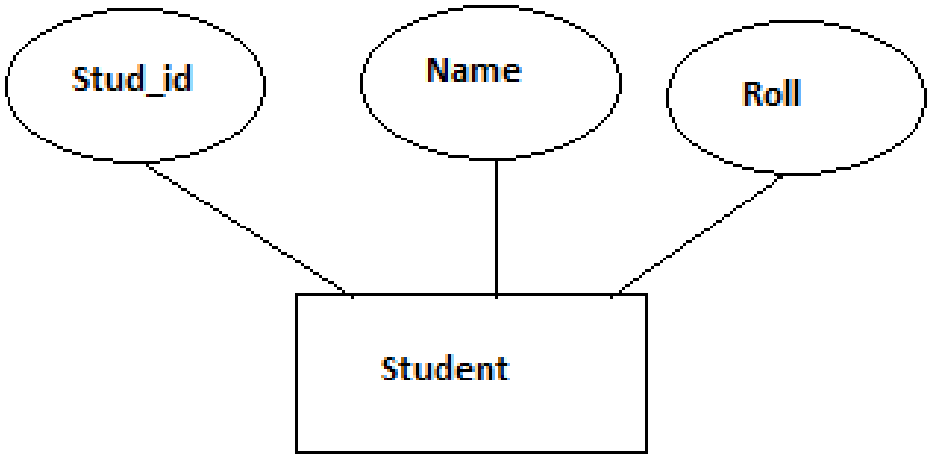



**Fig. 3.4**

**Single valued versus multivalued :**

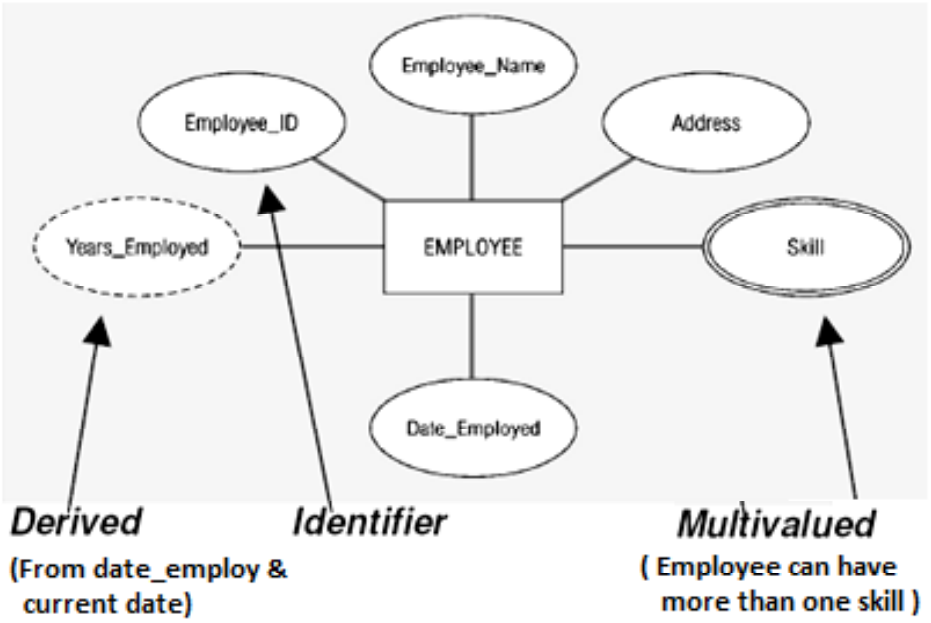



**Fig. 3.5**

**Single valued :** Single value associated with an attribute. Multivalued: may have more than one values. E.g., University degree attribute may contain B.Eng., M.Eng., or Ph.D.

**Degree of the relationship sets :** Refers to number of entity sets that participate in a relationship set. Relationship sets that involve two entity sets are binary (or degree two). Generally, most relationship sets in a

database system are binary. Relationship sets may involve more than two entity sets. The entity sets customer, loan, and branch may be linked by the ternary (degree three) relationship set. Or may say: Urnary Relationship, Binary relationship, Ternary Relationship.

**Unary Relationship :**

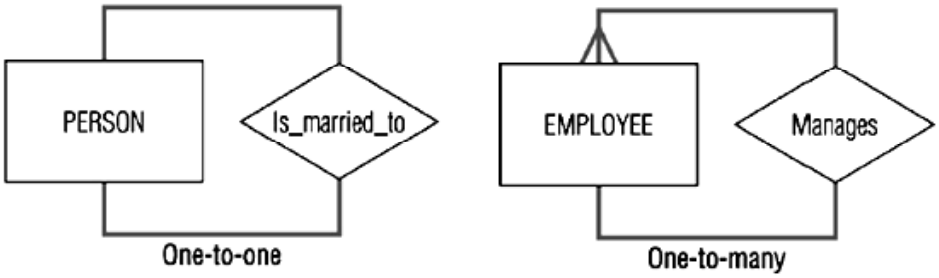



**Fig. 3.6**

**Binary relationship :**

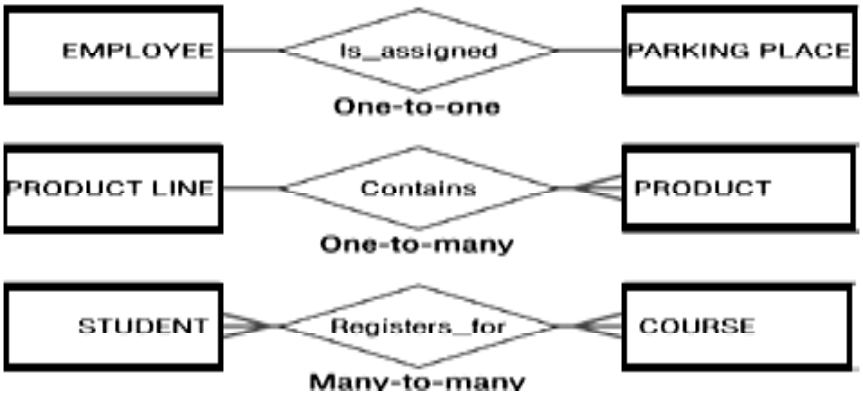



**Fig. 3.7**

**Ternary relationship :**

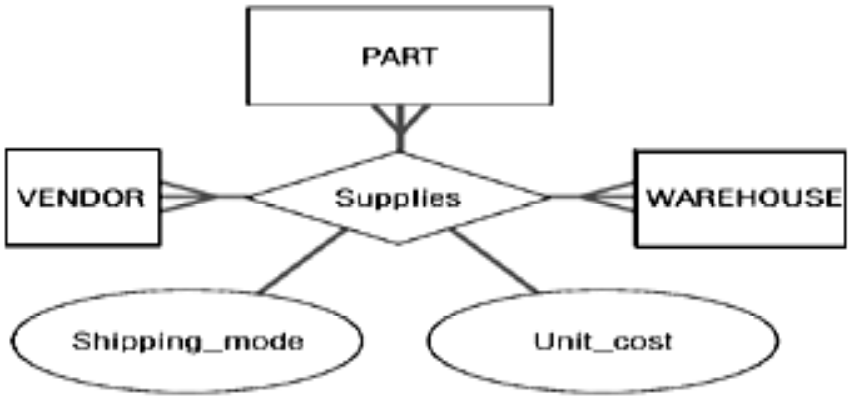



**Fig. 3.8**

Example of Binary Relationship: works_for relationship between Employee entity type and Department entity type.

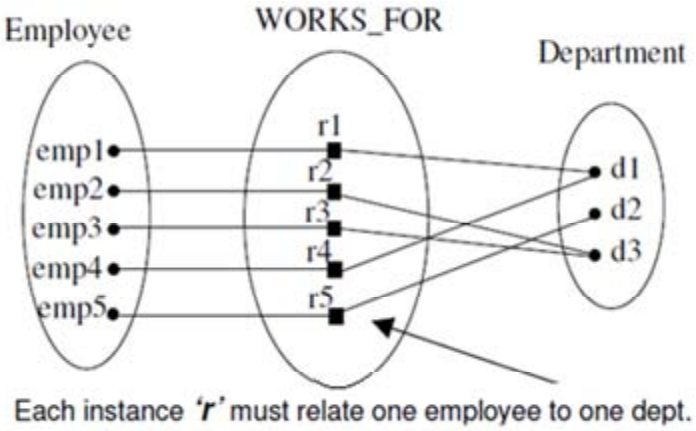



**Fig. 3.9**

**Example of Ternary Relationship :** Assigned_to relationship between Employee entity type ,Project entity type and Department entity type.

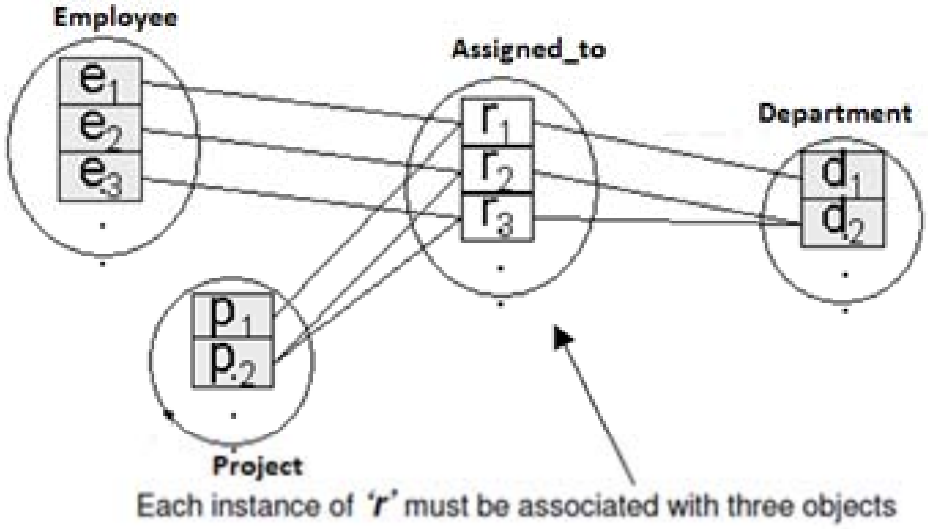



**Fig. 3.10**

**Weak Entity Sets :** A weak entity is an entity that cannot be uniquely identified by its own attributes alone.An entity set that does not have a primary key is referred to as a weak entity set. The existence of a weak entity set depends on the existence of a strong entity set; it must relate to the strong set via a one-to-many relationship set.

The discriminator (or partial key) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set. The

primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

**Example :**

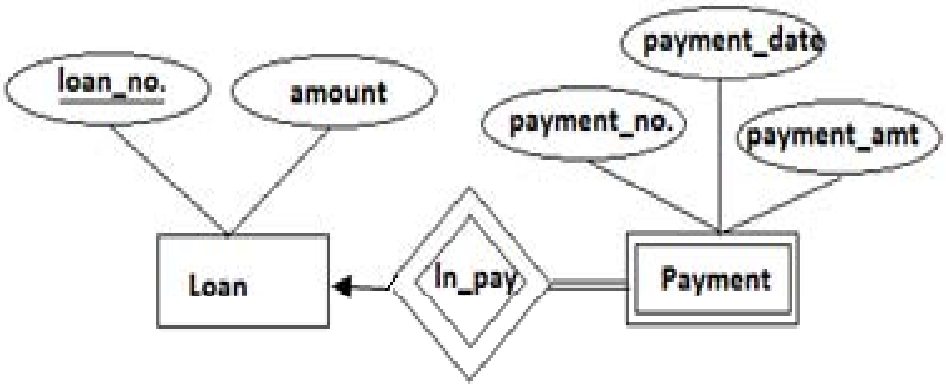



**Fig. 3.11**

The entity set payment has a attribute payment_no. , which is a sequence of numbers,starting from 1,generated separately for each loan.Thus although each payment entity is distinct,payments for different loans may share the same payment_no. .Thus ,this entity set doesnot have a primary key;it is weak entity set.

**E-R Diagram for library management system :** In the library Management system, the following entities and attributes can be identified.

1. Book -the set all the books in the library. Each book has a Book-id, Title, Author, Price, and Available (y or n) as its attributes.
2. Member-the set all the library members. The member is described by the attributes Member_id, Name, Street, City, Zip_code, Mem_type, Mem_date (date of membership), Expiry_date.
3. Publisher-the set of all the publishers of the books. Attributes of this entity are Pub_id, Name, Street, City, and Zip_code.
   Supplier-the set of all the Suppliers of the books. Attributes of this entity are Sup_id, Name, Street, City, and Zip_code.

**Assumptions :** a publisher publishes a book. Supplier supplies book to library. Members borrow the book (only issue).

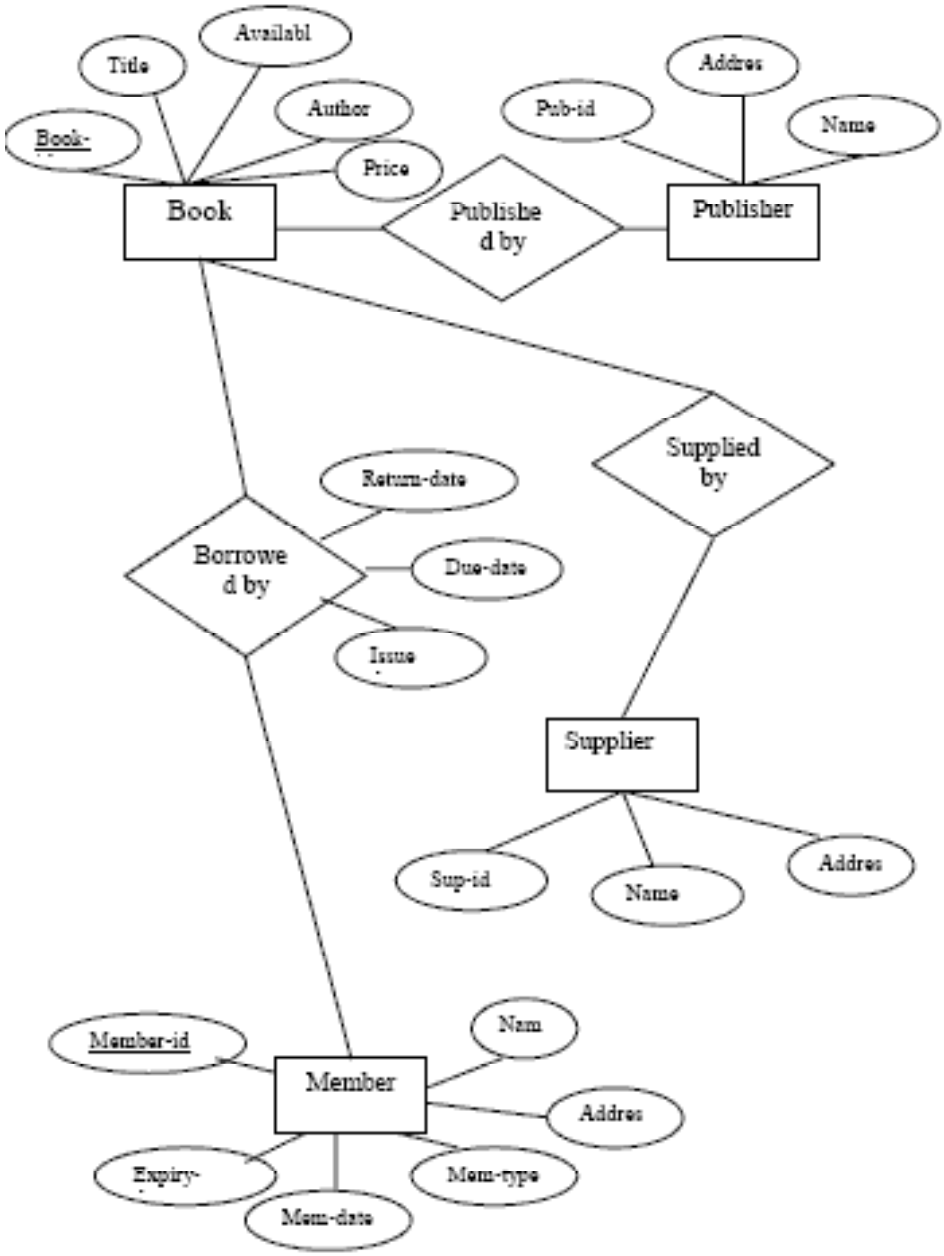



Fig. 3.12 : E-R Diagram of Library Management System

**Example 2 :** Say we are given a task of designing a database for an university system. We try to recognise the various entities that form a university sytem and then establish relationships among them.

Each entity is shown as a rectangle. For weak entities the rectangle has a double border. In the above diagram regular entities are University, College, Dean, Professor, Department, Student and Course. Section is a weak entity.

Properties or attributes of an entity are shown in ellipses and are attached to their respective entity by a single solid line. In this diagram I am showing properties for only student entity for the sake of clarity of the diagram.

The relationship between entities are shown as diamonds and the entities which are a part of the relationship are connected to the diamond by a solid line labeled either '1' or 'M' indicating whether the relationship is one-to-many, one-to-one or many-to-many.

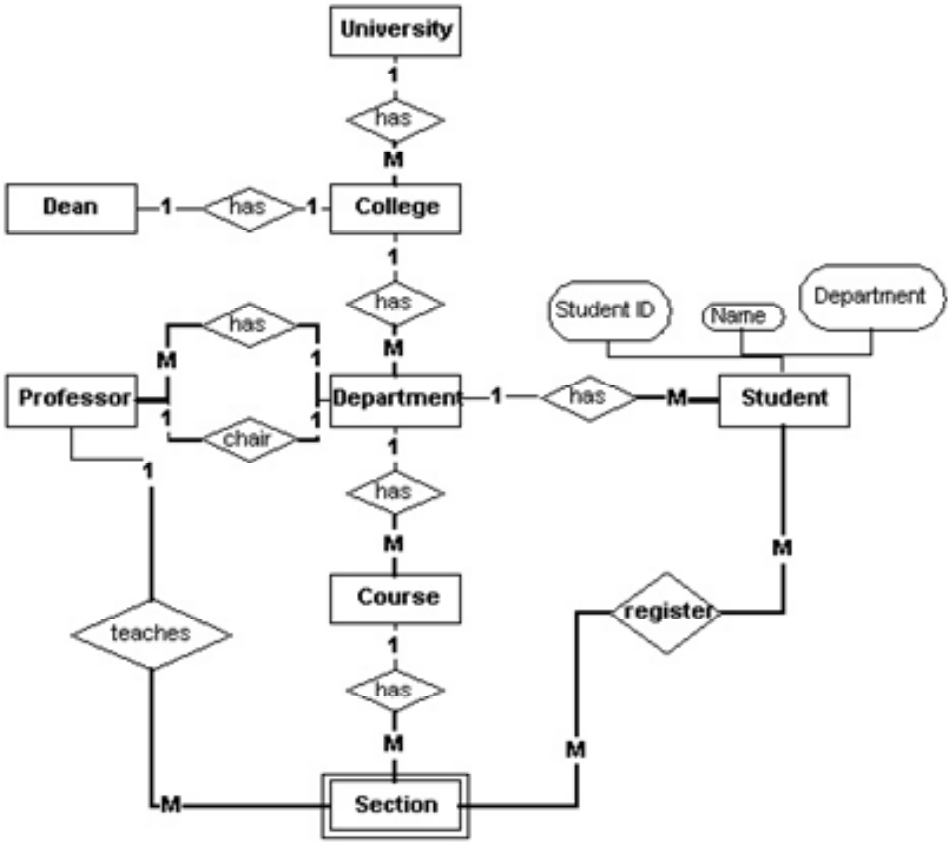
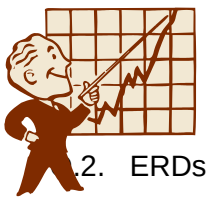



**Fig. 3.13**



**CHECK YOUR PROGRESS**

2. ERDs provide a description of :

a) The processes that occur in the model

b) The various entities and their relationships in the model

c) The entities in the model

d) The processes and data requirements of a model

e) The User Interface aspects of the model

Q.3. Which one of the following statements is true (select one)?

a) An ERD can only display entity type names.

b) Attributes are a rarely considered aspect of entities.

c) Attributes must always be displayed in ERDs.

d) Attributes equate to table names in a database.

e) Attributes can optionally be displayed in ERDs.

Q4. Fill in the blanks :

i) An instance is _____.

a) a set of relationship b) set of attributes

c) set of entites d) schema

ii) ER Model is used in _____ phase

a) Conceptual database b) Schema refinement

c) Physical refinement d) Applications and security

iii) The owner entity and weak entity set should participate in _____

a) many to many relationship

b) one to many relationship

c) many to one relationship

d) one to One relationship

iv) _____ entities are entities that cannot exist except with an identifying relationship with a regular entity type.

a) Composite b) Weak

c) Regular d) Associative

## 3.8 EXTENDED ER DIAGRAM

**SPECIALIZATION :** Specialization is process of defining a set of subclasses of an entity type.It is a set of subclasses form a specialization on the basis of some distinguishing characteristics. It employsTop-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.These subgroupings become lower-

level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set. It define different roles for different entities of same entity type and associate additional specific attributes to subclass. Depicted by a triangle component labeled ISA (i.e., savings-account is an account).

**GENERALIZATION :** Specialization and generalization are simple inversions of each other, they are represented in an E-R diagram in the same way.A bottom-up design process that combine a number of entity sets that share the same features into a higher-level entity set to generalize a set of entity types into a single superclass . Generalization consists of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common characteristics. Typically the subclasses are defined first, the superclass is defined next, and any relationship sets that involve the superclass are then defined.

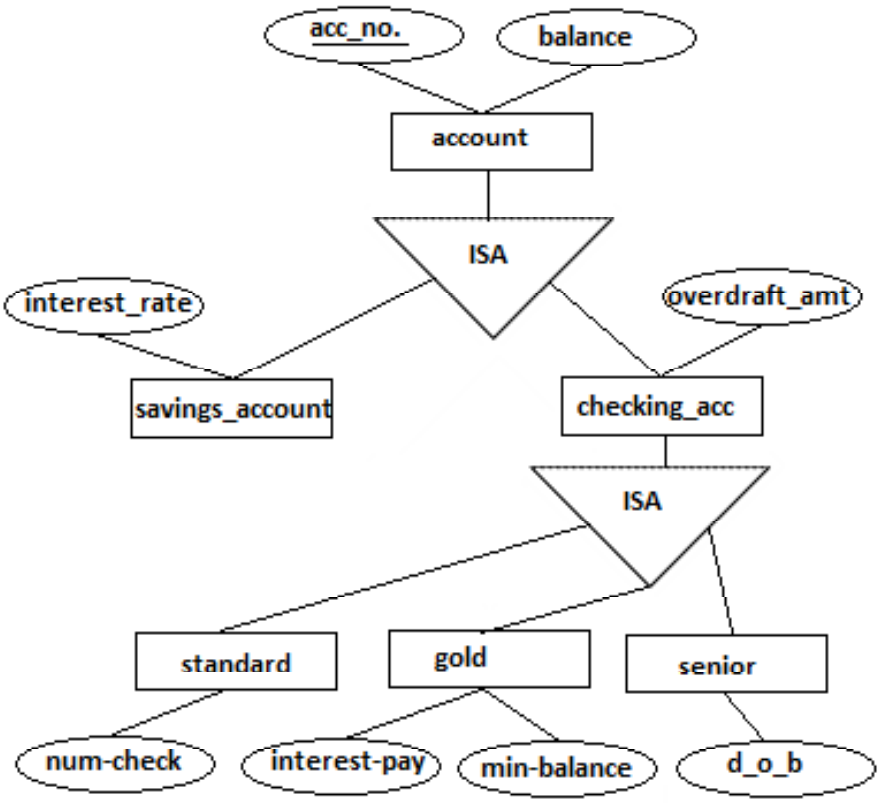



**Fig. 3.14**

We can specify two kinds of constraints with respect to ISA

hierarchies, namely, *overlap* and *covering* constraints.

**Overlap constraints** determine whether two subclasses are allowed to contain the same entity.

**Covering constraints** determine whether the entities in the subclasses collectively include all entities in the superclass.

Specialization and Generalization depicted by triangle component labeled ISA.

**AGGREGATION :** It allows us to indicate that a relationship set identfied through a dashed box participates in another relationship set. This is illustrated in Figure below, with a box around Sponsors and its participating entity sets used to denote aggregation. This effectively allows us to treat Sponsors as an entity set for purposes of defining the Monitors relationship set.We use aggregation when we need to express a relationship among relationships.

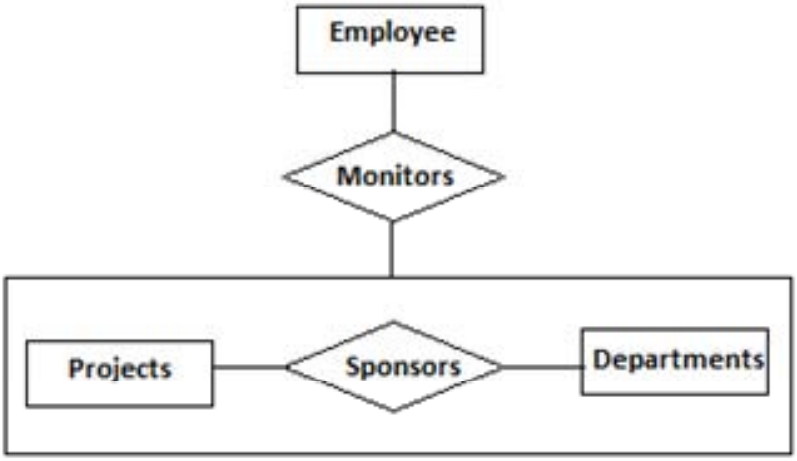



**Fig. 3.13**

**CHECK YOUR PROGRESS**

Q.5. Select the correct answer :

i) In a Hierarchical model records are organized as

A) Graph B) List C) Links D) Tree.

ii) In an E-R diagram attributes are represented by

A) rectangle B) square.

C) ellipse D) triangle.

iii) A relational database developer refers to a record as

A) a criteria B) a relation.

C) a tuple D) an attribute.

iv) SET concept is used in :

A) Network Model B) Hierarchical Model

C) Relational Model D) None of these

v) What is a relationship called when it is maintained between two entities?

A) Unary B) Binary

C) Ternary D) Quaternary

vi) Which of the following is record based logical model?

A) Network Model B) Object oriented model

C) E-R Model D) None of these

vii) Hierarchical model is also called

A) Tree structure B) Plex Structure

C) Normalize Structure D) Table Structure

## 3.9 LET US SUM UP

- A model is a representation of reality, 'real world' objects and events,and their associations.
- Three types odf data models-Record Based Data Models,Object Based Data Models,Physical Data Models
- Types of record based data model-Hierarchical data model, Network data model.Relational data model.
- The hierarchical data model organizes data in a tree structure. There is a hierarchy of parent and child data segments.
- The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type.
- Relational Model is a clean and simple model that uses the concept

of a relation using a table rather then a graph or shapes.

- A physical data model is a representation of a data design which takes into account the facilities and constraints of a given data base management system.
- An entity is an object that exists and is distinguishable from other objects.
- A relationship is an association between several entities. A relationship set is a set of relationships of the same type.
- ER diagram is a data modeling technique that creates a graphical representation of the entities, and the relationships between entities, within an information system.
- Attributes are the data we collect about the entities.
- Specialization is process of defining a set of subclasses of an entity type.It is a set of subclasses form a specialization on the basis of some distinguishing characteristics.
- Specialization and generalization are simple inversions of each other, they are represented in an E-R diagram in the same way.
- AGGREGATION: It allows us to indicate that a relationship set identfied through a dashed box participates in another relationship set.

## 3.10 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1 :** i) A, ii) B, iii) B, iv) C, v) C, vi) C

**Ans. to Q. No. 2 :** (b)

**Ans. to Q. No. 3 :** (e)

**Ans. to Q. No. 4 :** i) b), ii) a), iii) b), iv) b)

**Ans. to Q. No. 5 :** i) D, ii) C, iii) C, iv) A, v) B, vi) A, vii) A

## 3.11 FURTHER READINGS


- *Database System Concepts* By Henry korth and A.Silberschatz.
- *An Introduction to Database System* by Bipin Desai.
- *File Structure* by Michael J. Folk, Greg, Riccardi.

## 3.12 MODEL QUESTIONS

Q.1. Explain the difference between weak and strong entity set.

Q.2. Draw an ER diagram for the following application from the manufacturing industry:

Each supplier has a unique name.

More than one supplier can be located in the same city.

Each part has a unique part number.

Each part has a colour.

A supplier can supply more than one part.

A part can be supplied by more than one supplier.

A supplier can supply a fixed quantity of each part.

Q.3. Define the concept of aggregation. Give an example where this concept is useful.

Q.4. Construct an ER Diagram . A General Hospital consists of a number of specialized wards (such as Maternity, Paediatry, Oncology, etc). Each ward hosts a number of patients, who were admitted on the recommendation of their own GP and confirmed by a consultant employed by the Hospital. On admission, the personal details of every patient are recorded. A separate register is to be held to store the information of the tests undertaken and the results of a prescribed treat ment. A number of tests may be conducted for each patient. Each patient is assigned to one leading consultant but may be examined by another doctor, if required. Doctors are spe cialists in some branch of medicine and may be leading con sultants for a number of patients, not necessarily from the same ward.

# UNIT 4 : RELATIONAL DATABASES

## UNIT STRUCTURE



## 4.1 LEARNING OBJECTIVES

After going through this unit, you will able to :

- learn about relational model
- define relation, tuple, attribute, cardinality, degree, domain
- define types of keys
- define Relational Algebra
- learn about the types of operations on Relational Algebra

## 4.2 INTRODUCTION

A relational database matches data by using common characteristics found within the data set. The resulting groups of data uses the relational model, a technical term for this is schema.

The software used in relational database is called a ***Relational Database Management System*** (RDBMS). The term "relational database" often refers to RDBMS software, not the database itself. Strictly, a relational database is a collection of relations (frequently called tables). Other items are frequently considered part of the database, as they help to organize and

structure the data, in addition to forcing the database to conform to a set of requirements.

Relational databases, as implemented in relational database management systems, have become a predominant choice for the storage of information in new databases used for financial records, manufacturing and logistical information, personnel data and much more. Relational databases have often replaced legacy hierarchical databases and network databases because they are easier to understand and use, even though they are much less efficient. As computer power has increased, the inefficiencies of relational databases, which made them impractical in earlier times, have been outweighed by their ease of use. However, relational databases have been challenged by Object Databases, which were introduced in an attempt to address the object-relational impedance mismatch in relational database, and XML databases.

## 4.3 RELATIONAL DATA MODEL

The relational model was introduced by *E.F. Codd* in 1969 as a way to make database management systems more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory.

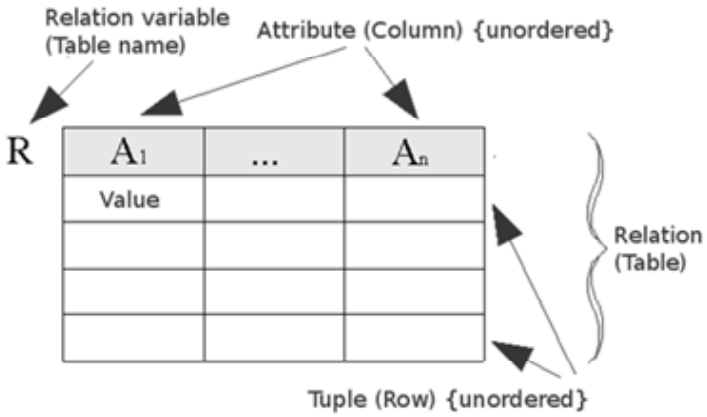
The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. Three key terms are used extensively in relational database models: relations, attributes, and domains. A relation is a table with columns and rows. The named columns of the relation are called attributes, and the domain is the set of values the attributes are allowed to take.

The basic data structure of the relational model is the table, where information about a particular entity say, an employee is represented in rows also called tuples and columns. Thus, the "relation" in "relational database" refers to the various tables in the database; a relation is a set of tuples. The columns enumerate the various attributes of the entity the employee's name, address or phone number, for example, and a row is an

actual instance of the entity a specific employee that is represented by the relation. As a result, each tuple of the employee table represents various attributes of a single employee.

All relations (i.e.,tables) in a relational database have to adhere to some basic rules to qualify as relations. First, the ordering of columns is immaterial in a table. Second, there cannot be identical tuples or rows in a table and third, each tuple will contain a single value for each of its attributes.

Relational Model concept :




Example of relational Model :

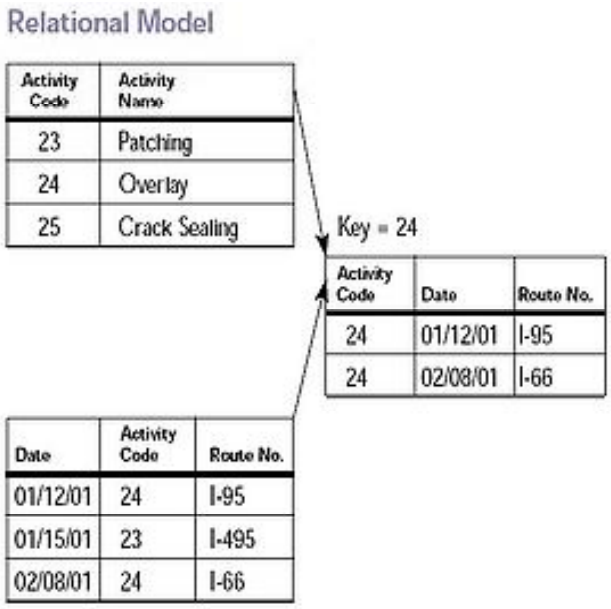



Fig. 4.2

One of the strengths of the relational model is that, in principle, any value occurring in two different records belonging to the same table or to different tables, implies a relationship among those two records. Yet, in order

to enforce explicit integrity constraints, relationships between records in tables can also be defined explicitly, by identifying or non-identifying parent-child relationships characterized by assigning cardinality . Tables can also have a designated single attribute or a set of attributes that can act as a "key", which can be used to uniquely identify each tuple in the table.

A key that can be used to uniquely identify a row in a table is called a primary key. Keys are commonly used to join or combine data from two or more tables. For example, an Employee table may contain a column named Location which contains a value that matches the key of a Location table. Keys are also critical in the creation of indexes, which facilitate fast retrieval of data from large tables. Any column can be a key, or multiple columns can be grouped together into a compound key. It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one.

Relational databases are currently the predominant choice in storing data like financial records, medical records, personal information and manufacturing and logistical data.The major advantages of the relational model over the older data models are its simple data representation and the ease with which even complex queries can be expressed.

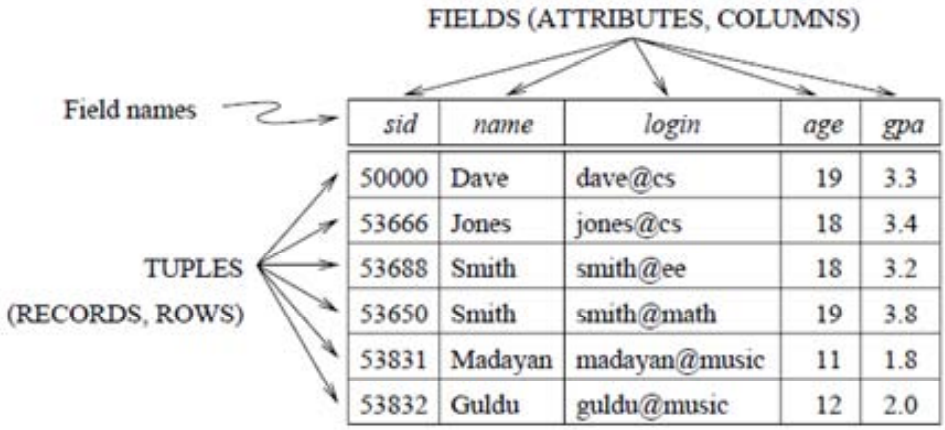
## 4.4 RELATION, TUPLE, ATTRIBUTE, CARDINALITY, DEGREE, DOMAIN

Let us consider a example of a relation schema:

Students(sid: string,
name: string,
login: string,
age: integer,
gpa: real )

Here we can see that the fields *name* and *sid* has a domain name *string*. The set of values associated with domain string is the set of all character strings.

An instance of a relation is a set of tuples, also called ***records***, in which each tuple has the same number of fields as the ***relation schema***. A relation instance can be thought of as a table in which each tuple is a ***row***, and all rows have the same number of fields. An instance of the *Students* relation appears in the figure.




| *sid* | *name* | *login* | *age* | *gpa* |
|---|---|---|---|---|
| 50000 | Dave | dave@cs | 19 | 3.3 |
| 53666 | Jones | jones@cs | 18 | 3.4 |
| 53688 | Smith | smith@ee | 18 | 3.2 |
| 53650 | Smith | smith@math | 19 | 3.8 |
| 53831 | Madayan | madayan@music | 11 | 1.8 |
| 53832 | Guldu | guldu@music | 12 | 2.0 |

**Fig. 4.3**

The instance contains six tuples and has five fields. Note that no two rows are identical. This is a requirement of the relational model,each relation is defined to be a *set* of unique tuples or rows. The order in which the rows are listed is not important.

A relation schema species the domain of each field or column in the relation instance. These domain constraints in the schema specify an important condition that we want each instance of the relation to satisfy: The values that appear in a column must be drawn from the domain associated with that column. Thus, the domain of a field is essentially the type of that field, in programming language terms, and restricts the values that can appear in the field.

The ***degree*** of a relation is the number of fields. The ***cardinality*** of a relation instance is the number of tuples in it. In the figure above(Fig.4.3) the degree of the relation (the number of columns) is five, and the cardinality of this instance is six. A relational database is a collection of relations with distinct relation names. The relational database schema is the collection of schemas for the relations in the database. For example, a University database with relations called Students, Faculty, Courses, Rooms, Enrolled,

Teaches.

An instance of a relational database is a collection of relation instances, one per relation schema in the database schema; of course, each relation instance must satisfy the domain constraints in its schema.

## 4.5 TYPES OF KEYS

***Keys*** are, as their name suggests, a key part of a relational database and a vital part of the structure of a table. They ensure each record within a table can be uniquely identified by one or a combination of fields within the table. They help enforce integrity and help identify the relationship between tables. There are three main types of keys, candidate keys, primary keys and foreign keys.

**Primary key :** A primary key uniquely defines a relationship within a database. In order for an attribute to be a good primary key it must not repeat. While natural attributes are sometimes good primary keys, surrogate keys are often used instead. A surrogate key is an artificial attribute assigned to an object which uniquely identifies it ,for instance, in a table of information about students at a school they might all be assigned a student ID in order to differentiate them. The surrogate key is useful through its ability to uniquely identify a tuple.

Another common occurrence, especially in regards to N:M cardinality is the composite key. A composite key is a key made up of two or more attributes within a table that (together) uniquely identify a record. For example, in a database relating students, teachers, and classes. Classes could be uniquely identified by a composite key of their room number and time slot, since no other class could have exactly the same combination of attributes. In fact, use of a composite key such as this can be a form of data verification, albeit a weak one.

A primary key is a column (or columns) in a table that uniquely identifies the rows in that table.

**Example :**

| Employee id | Name | Designation |
|---|---|---|
| 123 | Ram | Manager |
| 124 | Mohan | Clerk |
| 125 | Sham | Clerk |
| 126 | ravi | Peon |

In the table above, *Employeeid* is the primary key.

**Foreign Key :** A foreign key is a reference to a key in another relation, meaning that the referencing table has, as one of its attributes, the values of a key in the referenced table. A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second. In other words, if we had a table A with a primary key X that linked to a table B where X was a field in B, then X would be a foreign key in B.

Foreign keys need not have unique values in the referencing relation. Foreign keys effectively use the values of attributes in the referenced relation to restrict the domain of one or more attributes in the referencing relation.

A foreign key could be described formally as: "For all tuples in the referencing relation projected over the referencing attributes, there must exist a tuple in the referenced relation projected over those same attributes such that the values in each of the referencing attributes match the corresponding values in the referenced attributes."

An example might be a student table that contains the course_id the student is attending. Another table lists the courses on offer with course_id being the primary key. The 2 tables are linked through course_id and as such course_id would be a foreign key in the student table.

Student Table

| Student_id | FirstName | LastName | Course_id |
|---|---|---|---|
| 10001 | Ram | Narayan | CS1001 |
| 10002 | Shyam | Gupta | CS1001 |
| 10003 | Ravi | Gupta | CS1002 |

Course Table

| Course_id | Course_Name |
|---|---|
| CS1001 | DBMS |
| CS1002 | Compiler Design |

**Candidate key :** A candidate key is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data. Each table may have one or more candidate keys, but one candidate key is special, and it is called the primary key. This is usually the best among the candidate keys. When a key is composed of more than one column it is termed a composite key.

Also we can define a candidate key of a relation is a minimal superkey for that relation; that is, a set of attributes such that

1. the relation does not have two distinct tuples with the same values for these attributes .
2. there is no proper subset of these attributes for which (1) holds.

Example 1 : Consider a relation variable (relvar) R with attributes (A, B, C, D) that has only the following two legal values r1 and r2:

**r1**

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c2 | d1 |
| a2 | b1 | c2 | d1 |

**r2**

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b2 | c2 | d1 |
| a1 | b1 | c2 | d2 |

Here r2 differs from r1 only in the A and D values of the last tuple.

For r1 the following sets have the uniqueness property, i.e., there are no two distinct tuples in the instance with the same values for the attributes in the set:

{A,B}, {A,C}, {B,C}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}

For r2 the uniqueness property holds for the following sets;

{B,C}, {B,D}, {C,D}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}

Since superkeys of a relvar are those sets of attributes that have the uniqueness property for all legal values of that relvar and because we assume that r1 and r2 are all the legal values that R can take, we can determine the set of superkeys of R by taking the intersection of the two lists:

{B,C}, {A,B,C}, {A,B,D}, {A,C,D}, {B,C,D}, {A,B,C,D}

Finally we need to select those sets for which there is no proper subset in the list, which are in this case:

{B,C}, {A,B,D}, {A,C,D}

These are indeed the candidate keys of relvar R.

**Example 2:**

Student Table

| Student_id | First Name | Last Name | Course_id |
|---|---|---|---|
| 10001 | Ram | Narayan | CS1001 |
| 10002 | Shyam | Gupta | CS1001 |
| 10003 | Ravi | Gupta | CS1002 |

For the table above we have a student_id that uniquely identifies the students in a student table. This would be a candidate key. But in the same table we might have the student's FirstName and Last Name that also, when combined, uniquely identify the student in a student table. These would both be candidate keys.

In order to be eligible for a candidate key it must pass certain criteria.

- It must contain unique values
- It must not contain null values
- It contains the minimum number of fields to ensure uniqueness
- It must uniquely identify each record in the table

Once your candidate keys have been identified you can now select one to be your primary key .

**Super key :** A superkey is defined in the relational model of database organization as a set of attributes of a relation variable for which it holds that in all relations assigned to that variable, there are no two distinct tuples that have the same values for the attributes in this set. Equivalently a superkey can also be defined as a set of attributes of a relvar upon which all attributes of the relvar are functionally dependent.

The set of all attributes is a trivial superkey. If attribute set K is a superkey of relvar R, then at all times it is the case that the projection of R over K has the same cardinality as R itself.

Informally, a superkey is a set of attributes within a table whose values can be used to uniquely identify a tuple. A candidate key is a minimal set of attributes necessary to identify a tuple, this is also called a minimal superkey. For example, given an employee schema, consisting of the attributes employeeID, name, job, and departmentID, we could use the employeeID in combination with any or all other attributes of this table to uniquely identify a tuple in the table. Examples of superkeys in this schema would be {employeeID, Name}, {employeeID, Name, job}, and {employeeID, Name, job, departmentID} which is the trivial superkey.

In a real database we do not need values for all of those attributes to identify a tuple. We only need, per our example, the set {employeeID}. This is a minimal superkey – that is, a minimal set of attributes that can be used to identify a single tuple. So, employeeID is a candidate key.

Example :

Monarchs table

| Monarch Name | Monarch Number | Royal House |
|---|---|---|
| Edward | II | Plantagenet |
| Edward | III | Plantagenet |
| Richard | III | Plantagenet |
| Henry | IV | Lancaster |

listing out all the non-empty sets of attributes:

- {Monarch Name}
- {Monarch Number}
- {Royal House}

- {Monarch Name, Monarch Number}
- {Monarch Name, Royal House}
- {Monarch Number, Royal House}
- {Monarch Name, Monarch Number, Royal House}

Second, eliminating all the sets which do not meet superkey's requirement. For example, {Monarch Name, Royal House} cannot be a superkey because for the same attribute values (Edward, Plantagenet), there are two distinct tuples:

- (Edward, II, Plantagenet)
- (Edward, III, Plantagenet)

Finally, after elimination, the remaining sets of attributes are the only possible superkeys in this example:

- {Monarch Name, Monarch Number} (Candidate Key)
- {Monarch Name, Monarch Number, Royal House}

## CHECK YOUR PROGRESS

Q.1. Select the correct answer :

i) Which of the following fields in a student file can be used as a primary key?

a) class b) Social Security Number

c) GPA d) Major

ii) A tuple is also known as

a) table b) relation c) row d) field

iii) A field or a combination of fields that has a unique value is a

a) Secondary b) Composite

c) Primary d) Foreign

iv) Relational algebra is a _____.

a) Procedural language b) Object oriented language

c) Query language d) Structured language

v) The degree of a relation is

a) number of tuples b) number of rows

c) number of instances d)number of fields

vi) A candidate key of a relation is also called

a) minimal composite key b) minimal superkey

c) minimal alternate key d) minimal foreign key

vii) An instance of relational schema R (A, B, C) has distinct values of A including NULL values. Which one of the following is true?

a) A is a candidate key b) A is not a candidate key

c) A is a primary Key d) Both a) and c)

## 4.6 RELATIONAL ALGEBRA

Relational algebra received little attention outside of pure mathematics until the publication of E.F. Codd's relational model of data in 1970. Codd proposed such an algebra as a basis for database query languages.

Relational algebra is one of the two formal query languages associated with the relational model. Queries in algebra are composed using a collection of operators. A fundamental property is that every operator in the algebra accepts one or two relation instances as arguments and returns a relation instance as the result. This property makes it easy to compose operators to form a complex query relational algebra expression is recursively defined to be a relation, a unary algebra operator applied to a single expression, or a binary algebra operator applied to two expressions.

The basic operators of the algebra are selection, projection, union, cross-product, and difference.Each relational query describes a step-by-step procedure for computing the desired answer, based on the order in which operators are applied in the query. The procedural nature of the algebra allows us to think of an algebra expression as a plan, for evaluating a query.

Relational algebra is essentially equivalent in expressive power to relational calculus and thus first-order logic, this result is known as Codd's

theorem. You must be careful to avoid a mismatch, that may arise between the two languages since negation, applied to a formula of the calculus, constructs a formula that may be true on an infinite set of possible tuples, while the difference operator of relational algebra always returns a finite result. To overcome these difficulties, Codd restricted the operands of relational algebra to finite relations only and also proposed restricted support for negation (NOT) and disjunction (OR). In practice the restrictions have no adverse effect on the applicability of his relational algebra for database purposes.

The instance contains six tuples and has five fields. Note that no two rows are identical. This is a requirement of the relational model,each relation is defined to be a *set* of unique tuples or rows. The order in which the rows are listed is not important.

A relation schema species the domain of each field or column in the relation instance. These domain constraints in the schema specify an important condition that we want each instance of the relation to satisfy: The values that appear in a column must be drawn from the domain associated with that column. Thus, the domain of a field is essentially the type of that field, in programming language terms, and restricts the values that can appear in the field.

The degree, also called arity, of a relation is the number of fields. The cardinality of a relation instance is the number of tuples in it. In Figure above, the degree of the relation (the number of columns) is five, and the cardinality of this instance is six. A relational database is a collection of relations with distinct relation names. The relational database schema is the collection of schemas for the relations in the database. For example, a university database with relations called Students, Faculty, Courses, Rooms, Enrolled, Teaches.

An instance of a relational database is a collection of relation instances, one per relation schema in the database schema; of course, each relation instance must satisfy the domain constraints in its schema.

## 4.7 RELATIONAL ALGEBRA OPERATIONS

**Selection and Projection :**

Relational algebra includes operators to ***select*** rows from a relation ($\sigma$) and to ***project*** columns ($\pi$'). These operations allow us to manipulate data in a single relation. Consider the table given below.

Student Table

| Student_id | Marks |
|---|---|
| 10001 | 80 |
| 10002 | 90 |
| 10003 | 55 |
| 10004 | 50 |
| 10005 | 77 |

The expression $\sigma_{Marks>60}(Student)$ evaluates to the relation as shown below.

| Student_id | Marks |
|---|---|
| 10001 | 80 |
| 10002 | 90 |
| 10005 | 77 |

The subscript *Marks>60* specfies the selection criterion to be applied while retrieving tuples. The projection operator *'*allows us to extract columns from a relation; For example, the expression $\pi_{Student\_id}$(Student) evaluates to the relation shown below:

| Student_id |
|---|
| 10001 |
| 10002 |
| 10003 |
| 10004 |
| 10005 |

Since the result of a relational algebra expression is always a relation, we can substitute an expression wherever a relation is expected. For example

$$\pi_{Student\_id}(\sigma_{Marks>77}(Student))$$

evaluates to the realtion shown below

| Student_id |
|---|
| 10001 |
| 10002 |

The selection operator species the tuples to retain through a selection condition. The selection condition is a boolean combination (i.e., an expression using the logical connectives $\wedge$ and $\vee$) of terms that have the form attribute op constant or attribute1 op attribute2, where op is one of the comparison operators (< , > etc).

**Set Operations :**

The standard operations on sets are also available in relational algebra. These are : union ($\cup$), intersection ($\cap$), set-difference (–), and cross-product (X).

**Union :** R $\cup$ S returns a relation instance containing all tuples that occur in either relation instance R or relation instance S (or both). R and S must be unioncompatible, and the schema of the result is defined to be identical to the schema of R. Two relation instances are said to be union-compatible if the following conditions hold:

a) they have the same number of the fields, and

b) corresponding fields, taken in order from left to right, have the same domains. For example :

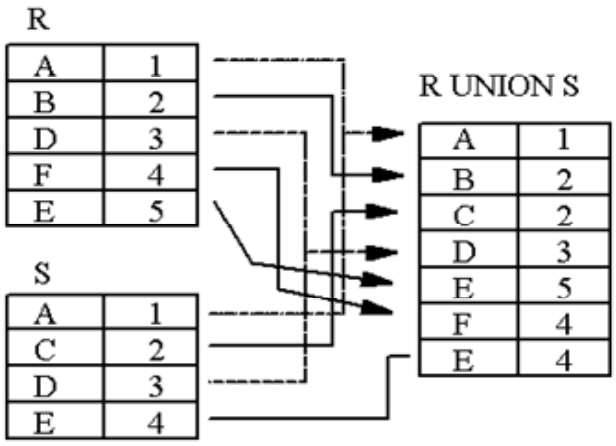



**Fig. 4.4**

**Intersection :** R ∩ S returns a relation instance containing all tuples that occur in both R and S. The relations R and S must be union-compatible, and the schema of the result is defined to be identical to the schema of R. For example

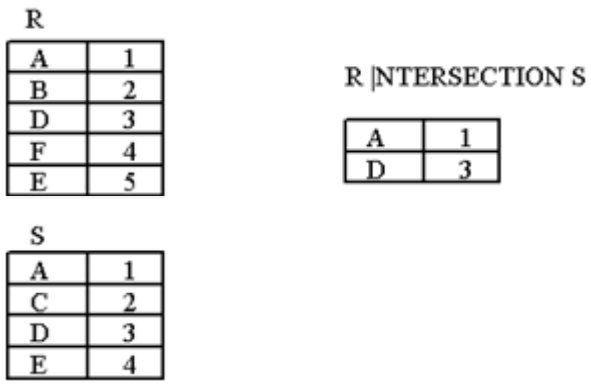



Fig. 4.5

**Set-difference :** R-S returns a relation instance containing all tuples that occur in R but not in S. The relations R and S must be union-compatible, and the schema of the result is dened to be identical to the schema of *R*. For example:

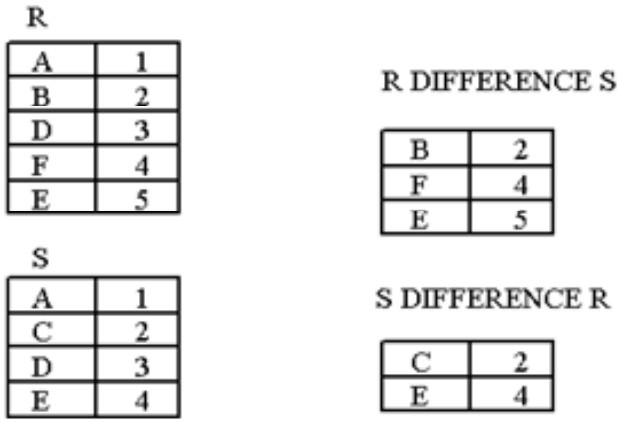



Fig. 4.6

**Cross-product :** R X S returns a relation instance whose schema contains all the fields of R (in the same order as they appear in R) followed by all the fields of S (in the same order as they appear in S). The result of R X S contains one tuple <r, s> (the concatenation of tuples r and s) for each pair of tuples r ε R; s ε S. The cross-product opertion is sometimes called Cartesian product.

We will use the convention that the elds of R X S inherit names

from the corresponding fields of R and S. It is possible for both R and S to contain one or more fields having the same name; this situation creates a naming conflict. The corresponding fields in R X S are unnamed and are referred to solely by position. For example:

R

| | |
|---|---|
| A | 1 |
| B | 2 |
| D | 3 |
| F | 4 |
| E | 5 |

S

| | |
|---|---|
| A | 1 |
| C | 2 |
| D | 3 |
| E | 4 |

R CROSS S

| | | | |
|---|---|---|---|
| A | 1 | A | 1 |
| A | 1 | C | 2 |
| A | 1 | D | 3 |
| A | 1 | E | 4 |
| B | 2 | A | 1 |
| B | 2 | C | 2 |
| B | 2 | D | 3 |
| B | 2 | E | 4 |
| D | 3 | A | 1 |
| D | 3 | C | 2 |
| D | 3 | D | 3 |
| D | 3 | E | 4 |

| | | | |
|---|---|---|---|
| F | 4 | A | 1 |
| F | 4 | C | 2 |
| F | 4 | D | 3 |
| F | 4 | E | 4 |
| E | 5 | A | 1 |
| E | 5 | C | 2 |
| E | 5 | D | 3 |
| E | 5 | E | 4 |

Fig. 4.7

**Natural Join :**

Invariably the JOIN involves an equality test, and thus is often described as an equi-join. Such joins result in two attributes in the resulting relation having exactly the same value. A `natural join' will remove the duplicate attributes.

R

| *ColA* | *ColB* |
|---|---|
| A | 1 |
| B | 2 |
| D | 3 |
| F | 4 |
| E | 5 |

S

| *SColA* | *SColB* |
|---|---|
| A | 1 |
| C | 2 |
| D | 3 |
| E | 4 |

R JOIN R.ColA = S.SColA S

| | | | |
|---|---|---|---|
| A | 1 | A | 1 |
| D | 3 | D | 3 |
| E | 5 | E | 4 |

R JOIN R.ColB = S.SColB S

| | | | |
|---|---|---|---|
| A | 1 | A | 1 |
| B | 2 | C | 2 |
| D | 3 | D | 3 |
| F | 4 | E | 4 |

Fig.4.8

- In most systems a natural join will require that the attributes have the same name to identify the attribute to be used in the join.

This may require a renaming mechanism.

- If you do use natural joins make sure that the relations do not have two attributes with the same name by accident.

**Outer Joins :**

Much of the data is lost when applying a join to two relations. In some cases this lost data might hold useful information. An outer join retains the information that would have been lost from the tables, replacing missing data with nulls.

The result of a join (or inner join) consists of tuples formed by combining matching tuples in the two operands, an outer join contains those tuples and additionally some tuples formed by extending an unmatched tuple in one of the operands by "fill" values for each of the attributes of the other operand.

There are three forms of the outer join, depending on which data is to be kept.

- LEFT OUTER JOIN - keep data from the left-hand table
- RIGHT OUTER JOIN - keep data from the right-hand table
- FULL OUTER JOIN - keep data from both tables

The result of the ***left outer join*** (represented by ⟕ )is the set of all combinations of tuples in R and S that are equal on their common attribute names, in addition to tuples in R that have no matching tuples in S. In the resulting relation, tuples in S which have no common values in common attribute names with tuples in R take a null value.For an example consider the tables Employee and Dept and their left outer join: NULL value is represented by ω.

***Employee***

| Name | EmpId | DeptName |
|---|---|---|
| Harry | 3415 | Finance |
| Sally | 2241 | Sales |
| George | 3401 | Finance |
| Harriet | 2202 | Sales |
| Tim | 1123 | Executive |

***Dept***

| DeptName | Manager |
|---|---|
| Sales | Harriet |
| Production | Charles |

*Employee ⋈ Dept*

| Name | EmpId | DeptName | Manager |
|---|---|---|---|
| Harry | 3415 | Finance | ω |
| Sally | 2241 | Sales | Harriet |
| George | 3401 | Finance | ω |
| Harriet | 2202 | Sales | Harriet |
| Tim | 1123 | Executive | ω |

Fig. 4.9

The result of the ***right outer join***(represented by ⟖ ) is the set of all combinations of tuples in R and S that are equal on their common attribute names, in addition to tuples in S that have no matching tuples in R.In the resulting relation, tuples in R which have no common values in common attribute names with tuples in S take a null value.For example consider the tables **Employee** and *Dept* and their right outer join:

*Employee*

| Name | EmpId | DeptName |
|---|---|---|
| Harry | 3415 | Finance |
| Sally | 2241 | Sales |
| George | 3401 | Finance |
| Harriet | 2202 | Sales |
| Tim | 1123 | Executive |

*Dept*

| DeptName | Manager |
|---|---|
| Sales | Harriet |
| Production | Charles |

*Employee ⟖ Dept*

| Name | EmpId | DeptName | Manager |
|---|---|---|---|
| Sally | 2241 | Sales | Harriet |
| Harriet | 2202 | Sales | Harriet |
| ω | ω | Production | Charles |

Fig. 4.10

The result of the ***full outer join***(represented by ⟗ ) is the set of all combinations of tuples in R and S that are equal on their common attribute names, in addition to tuples in S that have no matching tuples in R and tuples in R that have no matching tuples in S in their common attribute names.In the resulting relation, tuples in R which have no common values

in common attribute names with tuples in S take a null value.For an example consider the tables Employee and Dept and their full outer join:

*Employee*

| Name | EmpId | DeptName |
|---|---|---|
| Harry | 3415 | Finance |
| Sally | 2241 | Sales |
| George | 3401 | Finance |
| Harriet | 2202 | Sales |
| Tim | 1123 | Executive |

*Dept*

| DeptName | Manager |
|---|---|
| Sales | Harriet |
| Production | Charles |

*Employee ⋈ Dept*

| Name | EmpId | DeptName | Manager |
|---|---|---|---|
| Harry | 3415 | Finance | ω |
| Sally | 2241 | Sales | Harriet |
| George | 3401 | Finance | ω |
| Harriet | 2202 | Sales | Harriet |
| Tim | 1123 | Executive | ω |
| ω | ω | Production | Charles |

Fig. 4.11

## CHECK YOUR PROGRESS

Q.2. Select the correct answer :

i) In the relational modes, cardinality is termed as:

A) Number of tuples B) Number of attributes

C) Number of tables D) Number of constraints.

ii) Cartesian product in relational algebra is

A) a Unary operator B) a Binary operator

C) a Ternary operator D) not defined.

iii) In case of entity integrity, the primary key may be

A) not Null B) Null

C) both Null & not Null D) any value.

iv) Relational Algebra is

A) Data Definition Language

B) Meta Language

C) Procedural query Language

D) None of the above

v) Which of the following operation is used if we are interested in only certain columns of a table?

A) PROJECTION B) SELECTION

C) UNION D) JOIN

vi) The result of the UNION operation between R1 and R2 is a relation that includes

A) all the tuples of R1 B) all the tuples of R2

C) all the tuples of R1 and R2

D) all the tuples of R1 and R2 which have common columns

vii) Which of the operations constitute a basic set of operations for manipulating relational data?

A) Predicate calculus B) Relational calculus

C) Relational algebra D) None of the above

## 4.8 LET US SUM UP

- The relational model is very simple and elegant; a database is a collection of one or more relations, where each relation is a table with rows and columns.
- An instance of a relation is a set of tuples, also called records, in which each tuple has the same number of fields as the relation schema.
- The degree, also called arity, of a relation is the number of fields. The cardinality of a relation instance is the number of tuples in it.
- A primary key is a column in a table that uniquely identifies the rows in that table.

- A foreign key is generally a primary key from one table that appears as a field in another where the first table has a relationship to the second.
- A candidate key is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data.
- A superkey is a set of attributes within a table whose values can be used to uniquely identify a tuple.
- Relational algebra includes operators to *select* rows from a relation and to *project* columns. These operations allow us to manipulate data in a single relation.
- Standard operations on sets are also available in relational algebra are union , intersection , set-dierence, and cross-product.
- Set-difference : R-S returns a relation instance containing all tuples that occur in R but not in S.
- Cross-product: R X S returns a relation instance whose schema contains all the fields of R (in the same order as they appear in R) followed by all the fields of S (in the same order as they appear in S).

## 4.9 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1 :** i) B, ii) C, iii) B, iv) C, v) D, vi) B, vii) B

**Ans. to Q. No. 2 :** i) A, ii) B, iii) A, iv) C, v) A, vi) D, vii) C

## 4.10 FURTHER READINGS


- *Database System Concepts* By Henry korth and A.Silberschatz.
- *An Introduction to Database System* by Bipin Desai.
- *File Structure* by Michael J. Folk, Greg, Riccardi.

## 4.11 MODEL QUESTIONS

Q.1. Consider the following collection of relation schemes:

professor(profname, deptname)

department(deptname, building)

committee(commname, profname)

a) Find all the professors who are in any one of the committees that Professor Smith is in.

b) Find all the professors who are in at least all those committees that Professor Smith is in.

c) Find all the professors who are in exactly (i.e., no more and no less) all those committees that Professor Smith is in.

d) Find all the professors who have offices in at least all those buildings that Professor Smith has offices in.

Q.2. EMPLOYEE( ssn, firstname, lastname, title, salary, dept id, mgr id)

DEPARTMENT( dept id, departmentname, totalbudget)

Use the two relations given above to write the following queries using relational algebra:

a) Find the first and last name of managers such that none of his/her employees work in the same department as their boss (i.e. the department of all employees is different than the department of their boss).

b) Find the name of all departments that has no employee working in them.

Q.3. What is the minimum possible number of attributes in a key. What conditions must hold for there to be only this minimum number of attributes in a key?

Q.4. A relation has 4 attributes.

a) What is the minimum possible number of keys that it can have? What condition must hold for there being only n keys where n is the minimum possible for a 4 attribute relation?

b) What is the maximum possible number of superkeys it may have?

c) What is the maximum possible number of keys it may have?

Q.5. Can the union of two dierent keys of a relation be a key? Explain why/why not.

Q.6. Is the intersection of two superkeys necessarily a key? Explain why/ why not.

Q.7. Give a relational expression that calculates the relational intersection of two relations using only some of the operators from {$\sigma$, $\pi$ , U , - }

# UNIT 5 : SQL (PART I)

## UNIT STRUCTURE



## 5.1 LEARNING OBJECTIVES

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

- learn about SQL and its usefulness
- learn the basic commands and functions of SQL
- learn how SQL can be used for data administration and manipulation
- describe how SQL can be used to query a database for useful information

## 5.2 INTRODUCTION

We have already learnt in the previous unit about the concept of relational databases. We have also acquainted with the terms associated with a relation like tuple, attribute and also about the concepts of keys.

Structures Query Language (SQL) is one of the most popular and widely used language for retrieving and manipulating the database data. In this unit we will discuss the basics of SQL. Here we will learn how to create tables and assign data types to the table columns. Also we will learn how to insert data into the created tables and in what way the data can be manipulated and retrieved.

## 5.3 INTRODUCTION OF SQL

Structured Query Language (SQL) is a relational database query language. It is also called Structured English Query Language (SEQUEL). It is the standard command set used to communicate with the relational database management system (RDBMS). It defines the methods that are used to create and manipulate the relational databases.

It was introduced by IBM in the 1970's for IBM's prototype relational model DBMS, System R, under the name SEQUEL at IBM's San Jose Research facilities. In 1980 the language was renamed as SQL. ORACLE was the first commercial RDBMS that supported SQL. These two companies were followed by several other companies. It was declared as a standard by the American National Standard Institute (ANSI) and by the International Standards Organization (ISO) in 1986 and they called it SQL-86. Different versions of SQL were developed after that with some variations. The latest official version of SQL is SQL: 2008. SQL is pronounced as "**S-Q-L**" ("ess-cue-ell").

SQL is simple and interactive in nature. SQL is a both data definition and data manipulation language. The Data Definition Language (DDL) contains commands that can be used to create and destroy a database and its related objects. After defining the database structure with DDL, the users can use the Data Manipulation Language (DML) to insert, retrieve and modify the data in the database.

## 5.4 CHARACTERISTICS OF SQL

- Database Independent Language: SQL is an open language. Almost all DBMS vendors offer SQL. Regardless of the DBMS, the result of a SQL query will be the same. There are many versions of the SQL language. Most of the RDBMS have their own proprietary extensions in addition to the SQL standard. SQL is used in various application to interact with the DBMS like ORACLE, MySQL, MS Access, SQL Server etc.
- Cross-Platform Abilities : SQL can be used on different hardware platforms. The same SQL statement can be used on a PC, a server or a mainframe. An SQL-enabled database and its programs can be moved from one DBMS to another vendor's DBMS with a minimum effort.
- Easy to learn and use: SQL is easy to learn and understand as it resembles simple English sentences.
- Programming is less : SQL requires less programming to extract, manipulate and organize the data. The tables in SQL can be joined in ways that are not easily done using other programming languages. Also, SQL can be used from within most programming languages. Whether the coding is done in C++ or Java, the method of choice for accessing data is SQL.
- Speed: It reduces the time required for creating and maintaining systems.

## 5.5 BASIC STRUCTURE

The database in a relational database management system consists of a collection of database tables. The basic structure of an SQL expression consists of three clauses :

1. Select
2. From
3. Where

The ***select*** clause lists the desired attributes of the relation(s) in the result of the query.

The ***from*** clause lists the relation(s) to be scanned in the evaluation of the expression.

The **where** clause consists of a predicate involving attributes of the relations that appear in the from clause.

A typical SQL query has the following form :

**select** $A_1, A_2, \ldots\ldots, A_n$

**from** $r_1, r_2, \ldots, r_m$

**where** P

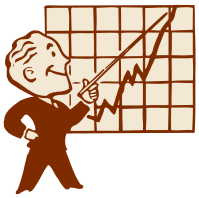
Each $A_i$ represents an attribute and each $r_i$ represents a relation. P is a predicate.

## 5.6 BASIC DATA TYPES

A data type is an attribute that specifies the type of data that the column in a table can hold. The following table shows the commonly used data type in SQL:

| Data Type | Description |
|---|---|
| CHAR(n) | Stores character strings of fixed length. The fixed size is specified in the parenthesis. It can hold upto 255 characters. The string can contain letters, numbers, and special characters. |
| VARCHAR2(n) | Stores character strings of variable length. The maximum size is specified in parenthesis. Can also store up to 255 characters. It is a more flexible form of the CHAR data type due to its variable length characteristics. |
| DATE | Stores date data. The standard format is DD-MON-YYYY. For example, 10-OCT-2008. DateTime stores date in the 24-hour format. |
| INT | Stores integers, that is numbers without a decimal part. |

| | |
|---|---|
| DECIMAL(p,q) | Sores a decimal number that is *p* digits long with *q* of these digits being decimal places to the right of the decimal point. For example, the data type DECIMAL(5,2) represents a number with three places to the left and two places to the right o the decimal (example 100.00) |



**CHECK YOUR PROGRESS**

Q.1. Fill up the blanks:

a) Structured Query Language is also called _____.

b) The first commercial RDBMS to support SQL was _____.

c) The _____ contains commands that can be used to create and destroy a database object.

d) The basic structure of an SQL expression consists of three clauses _____, _____ and _____.

e) The _____ is used to insert, retrieve and modify the data in the database.

f) The _____ lists the desired attributes of the relation(s) in the result of the query

g) The _____ datatype stores character strings of variable length.

h) The standard format of DATE datatype is _____.

i) _____ is the standard command set used to communicate with the relational database management system.

j) SQL was declared as a standard by the _____ and by the _____.

## 5.7 DDL COMMANDS

The Data Definition Language or Data Description Language (DDL) commands are used to define the structure or schema of a database. The term DDL was first introduced in relation to the Codasyl database model where the schema of the database was written in DDL. Some of the examples of DDL commands are as follows:

- **CREATE :** used to create objects in the database.
- **ALTER :** used to alter the structure of a database.
- **DROP :** used to delete objects from the database.
- **TRUNCATE :** used to remove all records from a database table.
- **RENAME :** used to rename an object.

**CREATE TABLE Command :** This is the most commonly used DDL command in SQL. The **CREATE TABLE** command defines the name of the table and the columns uniquely. It also specifies the type of data allowed in each of the column.

The syntax for creating SQL table is as follows:

**CREATE TABLE** <table_name>
(column1_definition,
column2_definition…
columnN_definition,
Primary key_definition,
Foreign key_definition
);

Each column definition has a minimum of three attributes – a name, datatype and size (or width) of the column. Each column definition is separated from the other by a comma. The Primary key list one or more columns that form the primary key. The foreign key is used to specify referential integrity constraints. The SQL statement is finally terminated with a semicolon.

**Rules for Creating tables in SQL :**

1. A name can have a maximum upto 30 characters.
2. Alphabets from A-Z, a-z and numbers from 0-9 are allowed.

3. A name should begin with an alphabet.
4. Use of special character like _ is allowed.

The figure below illustrates examples of creating tables for a store's database.

```
CREATE TABLE Employee(
        emp_id varchar2(10),
        emp_name varchar2(25),
        position varchar2(20),
        phone integer,
        sex varchar2(5),
        salary integer
);
CREATE TABLE Product(
        prod_id varchar2(10),
        prod_name varchar2(25),
        prod_qty integer
);
CREATE TABLE Customer(
        cust_id varchar2(10),
        cust_name varchar2(25),
        address varchar2(25),
        cust_phone integer,
        balance float
);
CREATE TABLE Sales(
        cust_id varchar2(10),
        prod_id varchar2(10),
        quantity integer);
```

**ALTER TABLE Command :** ALTER TABLE command is used to modify the structure of the tables in a database. It is used to add, delete columns of a table and to change the definition of an existing column .

**Adding new columns**, the syntax is as shown below:

**ALTER TABLE** <table name>

**ADD** (<new_column name1> <datatype> <(size) >,

<new_column name2> <datatype> <(size) >, …. );

Example: Adding a new column called 'address' in the table Employee, the command is as below:

**ALTER TABLE** Employee **Add** (address varchar2 (25));

**Dropping a column from a table**, the syntax is as shown below:

**ALTER TABLE** <table name> **DROP COLUMN** <columnname>;

*Example :* Drop the column cust_phone from the Customer table.

ALTER TABLE Customer DROP COLUMN cust_phone;

**Modifying existing column,** the syntax is as shown below:

**ALTER TABLE** <tablename>

**MODIFY**(<columnname> <new datatype>(<newsize>));

Example : Alter the Employee table so that the name field can hold maximum of 30 characters.

**ALTER TABLE** Employee **Modify** (emp_name varhar2(30));

The ALTER TABLE command has the following restrictions :

- It cannot change the table name
- It cannot change the name of a column
- If a table contains data in it, then this command cannot decrease the size of the column.

**DROP TABLE Command :** If a table in a database needs to be discarded then the DROP TABLE command is used. Dropping a table will automatically delete all the records entered into it.

The syntax for the command is as follows:

**DROP TABLE** <tablename>;

Example: Remove the Sales table with all the data in it.

**DROP TABLE** Sales;

**TRUNCATE TABLE Command :** The Truncate table command deletes all the records of a table. It empties the table completely. The syntax is as follows:

**TRUNCATE TABLE** <tablename>;

Example: Delete the records of Product table.

TRUNCATE TABLE Product;

**RENAME Command :** The RENAME command in SQL is used to rename a table. The syntax is as follows:

**RENAME** <TableName> **TO** <NewTableName>;

Example: Change the name of the Customer table to Cust.

RENAME Customer TO Cust;

**CHECK YOUR PROGRESS**

Q.2. State True or False :

i. The Data Definition Language (DDL) commands are used to define the structure of a database.

ii. The **CREATE TABLE** command along with defining the table name can also specify the data type allowed in each of the column.

iii. The ALTER TABLE can be used to change the table name.

iv. The column(s) of a table can be deleted using the DROP TABLE command.

v. The Truncate table command deletes all the records of a table.

Q.3. Fill in the blanks:

i. In each column definition, the minimum three attributes are _____, _____ and _____ of the column.

ii. To add and delete columns of a table, the command that is used is _____ command.

iii. Dropping a table by the DROP TABLE command will automatically delete all the _____ entered in it.

iv. To change the name of a table the _____ is used.

v. A table name should always begin with an _____.

## 5.8 DML COMMANDS

The Data Manipulation Language (DML) commands allow the user to make changes in the data of the database, that is, the contents of the database can be manipulated. Some of the examples of DML commands are as follows:

- **INSERT :** Inserts data into a table.
- **UPDATE :** Updates the existing data in a table.
- **DELETE :** It deletes all records of a table.

**INSERT Command :** The INSERT command is used to enter or insert rows (or records) into a table after it has been created with the help of the CREATE TABLE command. This command first creates a new empty row in the table, where the entered data will be inserted. *The syntax for the command is as follows:*

**INSERT INTO** <tablename> (<columnname1>, <columnname2>, …)

**VALUES** (<expression1>, <expression2>, ….);

It inserts a single row of data at a time.

Example: To add a new employee's details to the Employee table of the database.

**INSERT INTO** Employee (emp_id, emp_name, phone, sex, salary)

**VALUES**('e1', 'Ram', 9812012345, 'M', 30000);

This statement will insert a row at the bottom of the Employee table with the values that are given in the parenthesis. If the values are added corresponding to all the columns in the table, then the column list can be ignored. Thus the above example can be rewritten as follows:

**INSERT INTO** Employee **VALUES** ('e1', 'Ram', 9812012345, 'M', 30000);

In this above statement, the list of the column names has been ignored as the number and order of the attributes match the Employee table and the values given in the statement corresponds to that order.

**UPDATE Command :** This UPDATE statement is used to make changes or modify the data values in a database table. It updates the data values of the columns with the new values replacing their earlier values. The syntax for this command is as follows:

**UPDATE** <tablename>

**SET <**column_name1> = <value1>, <column_name2> = <value2>;

**WHERE** condition;

The WHERE clause given in the syntax above is optional. If a condition is not specified with the WHERE clause, then all the rows in the table will be updated.

Example 1: Increment the salary of all the Employees by 1000.

**UPDATE** Employee

**SET** salary = salary + 1000;

Example 2: Increase the salary of all the Employees, who gets less than Rs. 5000 by 1000.

**UPDATE** Employee

**SET** salary = salary + 1000

**WHERE** salary < 5000;

The less than sign (< ) used with the WHERE condition is known as the comparison operator. The table below lists all the comparison operators used in SQL.

**Table 5.0 : Comparison Operators**

| Operator | Description |
|---|---|
| = | Equal to |
| < | Less Than |
| > | Greater Than |
| <= | Less Than or Equal to |
| >= | Greater Than or Equal to |
| <> | Not Equal to |

**DELETE FROM Command :** The DELETE command is used to delete all records or selected records from a specified table. The syntax for the DELETE Command is as follows:

**DELETE**

**FROM** <tablename>

**WHERE** <condition>;

The WHERE clause is optional. If any condition is not specified with the WHERE clause, then this command will delete all the rows from the table.

Example 1: Delete all rows from the SALES table where the customer id is ‘c5’.

**DELETE**

**FROM** Sales

**WHERE** cust_id=’c5’;

Example 2: Delete all records of the Customer table.

**DELETE FROM** CUSTOMER;

## 5.9 DQL COMMANDS

The Data Query Language (DQL) is a type of SQL statement that allows retrieving or viewing the data in a database table. The **SELECT** statement of SQL is an example of a DQL command. The result of a SELECT statement is stored in a temporary table (relation), which is actually displayed. This statement allows retrieving records from one or more tables.

## 5.10 SELECT STATEMENT

The SELECT statement is the basic statement used to query the database. A query is used to extract data from the database. The SELECT statement is used in conjunction with the FROM clause to retrieve data from the database in an organized way.

**To display all the records of a table**, the syntax is as follows:

**SELECT * FROM** <tablename>;

The asterisk (*) is a wild card character that means everything. In the above statement, the asterisk (*) means list the data of all the column names from the specified table.

Example 1: Show the details of the Employees from the EMPLOYEE table.

```
SELECT * FROM Employee;
```

**To display some specific columns from a table**, the syntax is as shown below:

**SELECT** <column_name1>, <column_name2> **FROM** <tablename>;

Example 2: Show the id and names of all the employees.

```
SELECT emp_id, emp_name FROM Employee;
```

## 5.11 WHERE CLAUSE

To display records, from a table, that fulfils some specified criteria, the **WHERE** clause is used with the **SELECT** keyword.

**SELECT** <column_name (s)>
**FROM** <tablename>
**WHERE** <condition>;

The operators that are allowed in the WHERE clause are given in table 6.0.

Example 3: Show the records of the customers living in Guwahati.

```
SELECT *
FROM Customer
WHERE address='Guwahati';
```

## 5.12 USEFUL RELATIONAL OPERATORS

In a Relational Database environment, operators are needed to derive information from the data stored in the database tables. As the tables are set of rows and columns, the relational operators should operate on sets. That is why some of the classical set theory operators like UNION, INTERSECTION, EXCEPT and JOIN operators also show up as relational operators.

**UNION** operator : The UNION relational operator allows to join information from two or more tables that have the same structure. Tables with same structure means :

- The tables must have the same number of columns.
- The corresponding columns must have identical data types and lengths.

The syntax for the SQL UNION is as :

**SELECT** column_name(s**) FROM** table_name1

**UNION**

**SELECT** column_name(s) **FROM** table_name2

The union of two tables returns all the rows that appear in either table. The duplicate rows are eliminated. To allow the duplicate values the UNION ALL operator is used. The syntax of which is as :

**SELECT** column_name(s) **FROM** table_name1

**UNION ALL**

**SELECT** column_name(s) **FROM** table_name2

Suppose there are two tables called EMP and CUST having three columns each and also suppose the data types of each of the corresponding columns in the tables are the same. Suppose the table have the following data in them.

SELECT * FROM EMP;

| First Name | Last Name | Age |
|---|---|---|
| Manab | Nath | 32 |
| Rahul | Sarma | 40 |
| Binod | Moshahary | 35 |

SELECT * FROM CUST;

| First Name | Last Name | Age |
|---|---|---|
| Hemant | Rajkonwar | 36 |
| Manab | Nath | 32 |
| Manash | Kalita | 25 |

Now, the Union of the above two tables displays a virtual result table containing all rows of the first table as well as all the rows of the second table.

```
SELECT * FROM EMP
UNION
SELECT * FROM CUST;
```

The result table will be as follows:

| First Name | Last Name | Age |
|---|---|---|
| Binod | Moshahary | 35 |
| Hemant | Rajkonwar | 36 |
| Manab | Nath | 32 |
| Manash | Kalita | 25 |
| Rahul | Sarma | 40 |

**INTERSECTION** operator : *T*he SQL INTERSECT operator also operates on two tables but unlike the UNION operator, it returns only those rows that appear in both the tables. It removes duplicate rows from the final result table. The INTERSECT ALL operator does not remove duplicate rows from the result table. The syntax for the INTERSECT operator is as :

[SQL statement1]

**INTERSECT**

[SQL statement2]

For example,

```
SELECT * FROM EMP
INTERSECT
SELECT * FROM CUST;
```

The result table would look as follows:

| First Name | Last Name | Age |
|---|---|---|
| Manab | Nath | 32 |

**EXCEPT** operator : The SQL EXCEPT operator returns all rows from a database table that appears in the first table but that do not appear in the second table.

Example :

```
SELECT * FROM EMP
EXCEPT
SELECT * FROM CUST;
```

The output table would be as follows:

| First Name | Last Name | Age |
|---|---|---|
| Binod | Moshahary | 35 |
| Rahul | Sarma | 40 |

**JOIN** operator: The JOIN operator is a powerful relational operator which can combine data from multiple tables. Tables are joined on the columns that have the same data type and width. SQL supports different types of JOIN operations — INNER, OUTER and CROSS.

The **INNER JOIN** is also known as Equi Join. This is because the SELECT WHERE statement generally compares two columns of two different tables with the equivalence operator '='. The syntax for this type of Join is as follows:

```
SELECT column_name(s)
FROM table_name1
INNER JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

Suppose the PRODUCT table has the following data:

| Prod_id | prod_name | prod_qty |
|---|---|---|
| P1 | Shirt | 100 |
| P2 | Trouser | 100 |
| P3 | T-shirt | 80 |

SALES table:

| cust_id | Prod_id | quantity |
|---|---|---|
| C1 | P1 | 35 |
| C5 | P3 | 55 |

Now, to list the quantity of the products sold, the SQL statement would be as :

```
SELECT Product.prod_name, Sales.quantity
FROM Product
INNER JOIN Sales
ON Product.Prod_id=Sales.Prod_id
```

The result table would be as

| Prod_name | quantity |
|---|---|
| Shirt | 35 |
| T-shirt | 55 |

**OUTER JOIN** is similar to INNER JOIN but is more flexible in selecting the data from the tables. This type of Join is used to select data or

rows from the table on the left or right or both regardless of whether the other table has common values.

The syntax for LEFT Join is :

```
SELECT column_name(s)
FROM table_name1
LEFT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

Suppose to list all the products along with the number of quantities sold, the SQL statement would be as :

```
SELECT Product. Prod-_name, Product.prod_qty,
Sales.quantity
FROM Product
LEFT JOIN Sales
ON Product.Prod_id=Sales.Prod_id
```

The result table would be as :

| prod_name | prod_qty | quantity |
|---|---|---|
| Shirt | 100 | 35 |
| Trouser | 100 | |
| T-shirt | 80 | 55 |

The LEFT Join returns all the rows from the left table even if there are no matches in the right table.

The RIGHT Join returns all the rows from the right table even if there are no matches in the left table. The syntax is as follows:

```
SELECT column_name(s)
FROM table_name1
RIGHT JOIN table_name2
ON table_name1.column_name=table_name2.column_name
```

**CROSS JOIN** returns a Cartesian product. Cartesian product means it returns the number of rows that is equal to the product of all rows in both the tables being joined. That is, it combines every row from the left table with every row from the right table. For example, if the first table has 20 rows and the second table has 10 rows, the result will be 20 * 10, or 200

rows. This type of query takes a long time to execute. The pictorial representation of cross join syntax is as:

```
Select *
From table1
Cross join table2
```

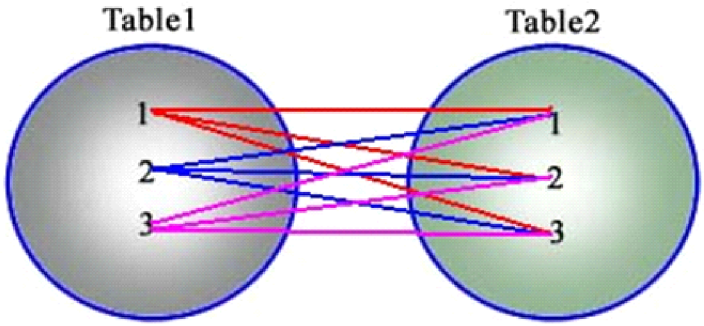



**Fig. 5.1**

## 5.13 AGGREGATE FUNCTIONS

The database tables created with SQL commands stores large amount of data. There are some aggregate functions in SQL to assist in the summarization of the large volumes of data. These aggregate functions in SQL returns a single value, calculated from the values in the column. Useful aggregate functions are as follows:

- AVG() ——-- Returns the average value
- COUNT() -- Returns the number of rows
- FIRST() -—- Returns the first value
- LAST() -—— Returns the last value
- MAX() -———-Returns the largest value
- MIN() -———- Returns the smallest value
- SUM() -—— Returns the sum

## 5.14 SUM FUNCTION

The **SUM()** function returns the total sum of the numeric values of a column. The syntax is as follows:

**SELECT SUM** <column_name> **FROM** <tablename>;

To understand this function better, let us take an example. Suppose we have the following data in the ***Products*** table.

| PROD_ID | PROD_NAME | PROD_QTY |
|---|---|---|
| P1 | T-Shirts | 200 |
| P2 | Jeans | 150 |
| P3 | Trousers | 100 |
| P4 | Pull Overs | 80 |
| P5 | Shirts | 200 |

**SELECT SUM** (prod_qty) "Total Quantity" **FROM** Product**;**

The result will be the total number of product quantities in the Product table. So, the output of the above statement will be

Total Quantity

**730**

## 5.15 AVG FUNCTION

The **AVG()** function calculates the average of the values in a specified column. The syntax is as follows:

**SELECT AVG** <column_name> **FROM** <tablename>;

Let us take an example. Suppose the Employee table contains the following data.

| Emp_id | Emp_name | Phone | Sex | Salary |
|---|---|---|---|---|
| E1 | Rani Patil | 9856723451 | F | 34000 |
| E2 | Nirmali Das | 9546751289 | F | 20000 |
| E3 | Binod Das | 8856359341 | M | 45000 |
| E4 | Manav | 9506781256 | M | 27000 |

**SELECT AVG** (salary) "Average" **FROM** Employee**;**

The output of the above statement will be as follows:

Average

**31500**

**CHECK YOUR PROGRESS**

Q.4. State True or False

i. The contents of a database can be manipulated with the help of the DML commands.

ii. To insert records into a table, the UPDATE command is used.

iii. The INSERT command is used only after the table has been created.

iv. If a condition is not specified with the WHERE clause in the UPDATE statement, then no rows in the table will be updated.

v. To delete all records or selected records from a specified table, the DELETE FROM command is used.

vi. To modify the values in a table, the Insert command is used.

vii. The SELECT statement is a DQL command.

viii. To count the number of rows in a table, the AVG() function is used.

ix. The SUM() function returns the total sum of the numeric values of a column.

x. The relational operators in SQL are used to combine data from different tables into one table.

## 5.16 LET US SUM UP

- Structured Query Language (SQL) is a relational database query language.
- ORACLE was the first commercial RDBMS that supported SQL.

- SQL was declared as a standard by the American National Standard Institute (ANSI) and by the International Standards Organization (ISO) in 1986.
- SQL is a both data definition and data manipulation language. The Data Definition Language (DDL) contains commands that can be used to create and destroy a database and the Data Manipulation Language (DML) contains commands that can be used to insert, retrieve and modify the data in the database.
- The basic structure of an SQL expression consists of three clauses — SELECT, FROM and WHERE.
- Some DDL commands are —— CREATE, ALTER, DROP, TRUNCATE AND RENAME.
- The **CREATE TABLE** command defines the name of the table and the columns uniquely. It also specifies the type of data allowed in each of the column.
- ALTER TABLE command is used to modify the structure of the tables in a database.
- DROP TABLE command deletes the table along with all the records in it.
- The TRUNCATE TABLE command empties the table data completely.
- To rename a table, the RENAME command is used.
- The Data Manipulation Language (DML) commands are INSERT, UPDATE and DELETE.
- To enter records in a table, the INSERT command is used.
- UPDATE statement is used to make changes or modify the data values in a database table. It updates the data values of the columns with the new values replacing their earlier values.
- The DELETE command is used to delete all records or selected records from a specified table.
- The SELECT statement is the basic statement used to query the database. It is a DQL command.

## 5.17 FURTHER READINGS


- *SQL for Dummies*, *Allen G. Taylor*, Wiley Publishing Inc.
- *Sams Teach yourself SQL in 21 days*, Fourth Edition by Sams publishing.
- *SQL, PL/SQL The Programming Language Of Oracle 4th Revised Edition*, Bayross, Ivan, Bpb publications.


## 5.18 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1 :** a) Structured English Query Language (SEQUEL).
b) ORACLE
c) Data Definition Language (DDL)
d) Select, From, Where
e) Data Manipulation Language (DML)
f) ***select*** clause
g) VARCHAR2
h) DD-MON-YYYY
i) SQL
j) American National Standard Institute (ANSI), International Standards Organization (ISO).

**Ans. to Q. No. 2 :** i) True, ii) True, iii) False, iv) false, v) true

**Ans. to Q. No. 3 :** i) name, datatype, width,
ii) ALTER TABLE
iii) records
iv) RENAME command
v) alphabet.

**Ans. to Q. No. 4 :** i. True, ii. False, iii. True, iv. False, v. True,
vi. False, vii. True, viii. False, ix. True, x. True

## 5.19 MODEL QUESTIONS

Q.1. What is SQL and what are the characteristics of SQL?

Q.2. What are the common data types used to define columns using SQL?

Q.3. Explain any three Data Definition Language commands.

Q.4. What is the difference between DROP TABLE command and TRUNCATE command?

Q.5. Explain the different variations of the Alter Table command. What are the restrictions on this command.

Q.6. Explain some of the Useful aggregate functions in SQL

# UNIT 6 : SQL (PART II)

## UNIT STRUCTURE



## 6.1 LEARNING OBJECTIVES

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

- learn how to manipulate the data in the database tables
- describe different types of operators used to handle the data
- illustrate the meaning of NULL values and also about how to handle them

## 6.2 INTRODUCTION

We have already learnt in the previous unit about the basics of Structured Query Language (SQL). We have already learnt the creation, insertion and deletion of tables and data of the tables. SQL can also execute queries against a database and can retrieve data from the databases. Updating of data in tables can also be performed in the tables.

In this unit we will discuss the operators that are used to operate the data in the database tables. Operators are used in almost every SQL statement. They are used to compare, evaluate or calculate values. They tell SQL how to evaluate an SQL expression or a conditional statement. We will find here that operators are mostly used in the WHERE clause of the SQL statement.

## 6.3 COMPOUND CONDITIONS AND LOGICAL OPERATORS

*Compound conditions* are simple conditions that are joined by logical operators. An *operator* is a symbol specifying an action that is performed on one or more expressions. Logical operators test for the truth of some condition. Most operators will appear inside conditional statements in the ***WHERE*** clause of SQL Commands. The logical operators in SQL are as follows:

AND, OR and NOT.

The syntax for a compound condition is as follows:

**SELECT** <column_name >
**FROM** <table_name >
**WHERE** <simple condition >
{[**AND|OR**] simple condition};

## 6.4 AND Operator

SQL ***AND*** joins together two or more conditions and returns result only when all of the conditions are true. **AND** helps to query for very specific

records. There is no limit to the number of **AND** conditions that can be applied to a query by utilizing the ***WHERE*** clause.

For example, the following query will find out only the rows where the customer identity is ‘c10’ and the product identity is ‘p5’. It will not find a row for customer identity ’c12’ and product identity p5.

```
SELECT *
FROM Sales
WHERE cust_id=’c10’
AND prod_id=’p5’;
```

The output of the above query will be as follows:

| Cust_id | Prod_id | Quantity |
|---|---|---|
| c10 | p5 | 25 |

This example illustrates how SQL ***AND*** combines multiple conditional statements into a single condition.

## 6.5 OR OPERATOR

The OR operator in SQL also connects two or more conditions but it returns results when any of the condition is true.

Suppose the ‘***Customers***’ table contains the following data in it:

| Cust_id | Cust_name | Address | Cust_phone | balance |
|---|---|---|---|---|
| C1 | John | Mumbai | 9876412345 | 0.00 |
| C2 | Rahul | Guwahati | 8812534675 | 0.00 |
| C3 | Jutika | Delhi | 9853510293 | 0.00 |
| C4 | Kamal | Guwahati | 8812356473 | 0.00 |
| C4 | Sona | Kolkata | 6723812345 | 0.00 |

To retrieve the customer names that stay either in Guwahati or in Delhi, the SQL statement is as follows:

```
SELECT cust_name
FROM Customer
WHERE address = ‘Guwahati’ OR address = ‘Delhi’;
```

The result of the above statement will be as :

| Cust_name |
|---|
| Rahul |
| Jutika |
| Kamal |

## 6.6 COMBINING "AND" AND "OR"S OPERATORS

The statements in SQL can be combined using the logical operators AND and OR. SQL engine will display the results when **all** the conditions specified with the AND operator are satisfied and when **any** of the conditions specified with the OR operator are specified. Let us take an example to understand how to combine these operators together into a single statement.

Suppose the '***Employee***' table contains the following data in it:

| emp_id | emp_name | position | phone | sex | salary |
|---|---|---|---|---|---|
| e1 | John | Manager | 9876412345 | M | 55000 |
| e2 | Rahul | Asst. Manager | 8812534675 | M | 34000 |
| e3 | Jutika | Manager | 9853510293 | F | 55000 |
| e4 | Kamal | AGM | 8812356473 | M | 80000 |
| e4 | Sona | Executive | 6723812345 | F | 20000 |

Fig : Table 1

Now, combining the AND and OR operators into a single statement,

```
SELECT emp_id, emp_name
FROM Employee
WHERE Position = 'Manager' AND sex='M'
OR Salary < 80000
```

This statement would return all those records where the employee's position is manager and is a male employee. It will also return those records where the salary of the employee is less than 80000.

## 6.7 IN Operator

The IN operator is a comparison operator. It is used to compare a value to a list of values that has been specified. It returns TRUE if the

compared value matches at least one of the values in the list. The syntax of the IN operator in SQL is as follows:

**SELECT** <column_name(s)>

**FROM<** table_name>

**WHERE** <column_name> **IN** (value1,value2,...);

Suppose we have the following data in the *PRODUCTS* table.

| PROD_ID | PROD_NAME | PROD_QTY |
|---|---|---|
| P1 | T-Shirts | 200 |
| P2 | Jeans | 150 |
| P3 | Trousers | 100 |
| P4 | Pull Overs | 80 |
| P5 | Shirts | 200 |

Suppose from the above table, we need the records of the product Id's P2, P4, P5. The SQL statement for this query is as follows:

**SELECT ***

**FROM** Products

**WHERE** prod_id **IN** ('P2,'P4','P5');

The result-set will look like as follows:

| PROD_ID | PROD_NAME | PROD_QTY |
|---|---|---|
| P2 | Jeans | 150 |
| P4 | Pull Overs | 80 |
| P5 | Shirts | 200 |

## 6.8 BETWEEN Operator

The BETWEEN operator is used with a WHERE clause to test whether a value lies in a specified range of values. x is BETWEEN a and b means that x >= a and x <= b. This is also a comparison operator. The syntax of BETWEEN operator is as follows:

**SELECT** <column_name(s)>

**FROM** <table_name>

**WHERE** <column_name>

**BETWEEN** value1 AND value2

Suppose the 'EMPLOYEE' table contains the following data in it:

| Emp_id | emp_name | Phone | Sex | Salary |
|---|---|---|---|---|
| E1 | Kabita | 9876412345 | F | 20000 |
| E2 | Babita | 8812534675 | F | 15000 |
| E3 | Neel | 9853510293 | M | 25000 |
| E4 | Konika | 8812356473 | F | 16000 |
| E5 | Manoj | 6723812345 | M | 17000 |

The following example shows all the Employee details that gets salary between Rs 16000 and Rs 25000.

**SELECT * FROM** Employee

**WHERE** Salary **BETWEEN** 16000 and 20000;

The result of the above query will be as given below:

| Emp_id | emp_name | Phone | Sex | Salary |
|---|---|---|---|---|
| E1 | Kabita | 9876412345 | F | 20000 |
| E3 | Neel | 9853510293 | M | 25000 |
| E4 | Konika | 8812356473 | F | 16000 |
| E5 | Manoj | 6723812345 | M | 17000 |

**CHECK YOUR PROGRESS**

Q.1. Fill up the blanks:

a) An _____ is a symbol specifying an action that is performed on one or more expressions.

b) Logical operators join the simple conditions in SQL to form _____.

c) AND operator returns result only when all of the conditions given in the SQL statement are _____.

d) The OR operator in SQL returns result only when _____ of the condition in the expression is true.

e) To compare a value to a list of values specified in an SQL expression, the _____ operator is used.

f) The IN operator returns _____ if the compared value matches at least one of the values in the list.

## 6.9 NOT Operator

The NOT operator is used to display only those records which do not satisfy the given conditions. Suppose from the EMPLOYEE table we want to list the details of the employees who are not male. The SQL statement for this query will be as follows:

```
SELECT *
FROM EMPLOYEE
WHERE NOT sex='M'
```

The result of the above statement will be a follows:

| Emp_id | emp_name | Phone | Sex | Salary |
|---|---|---|---|---|
| E1 | Kabita | 9876412345 | F | 20000 |
| E2 | Babita | 8812534675 | F | 15000 |
| E4 | Konika | 8812356473 | F | 16000 |

The NOT operator can also be used with a comparison operator to negate the result of the comparison.

```
NOT BETWEEN
NOT IN (value1, value2, value3,...)
NOT LIKE
```

## 6.10 ORDER OF PRECEDENCE FOR LOGICAL OPERATORS

Operator precedence determines the sequence in which the operations are performed in an SQL expression with multiple operators. While evaluating the SQL statements with multiple operators, the operators with higher precedence are evaluated first before evaluating those with lower precedence.

When more than one logical operators are present in an SQL expression, NOT is evaluated first, then AND and finally OR.

Order of Precedence for logical operators is shown in the table below. Arithmetic operators and comparison operators take higher precedence than logical operators.

| Order Evaluated | Operator |
|---|---|
| 1 | Arithmetic operators |
| 2 | Comparison operators |
| 3 | NOT |
| 4 | AND |
| 5 | OR |

## 6.11 LIKE Operator

The LIKE operator is used with a WHERE clause to compare a character string to a specified pattern. It also is a comparison operator. It searches for the specified pattern in a column of a database table. This is achieved by using wildcard characters. The two wildcard characters are —

**%** – allows to match strings of any length, that is zero or more characters.

_ – allows to match a single character.

The syntax for this operator is as follows:

**SELECT** <column names>

**FROM** <tablename>

**WHERE** <column_name> **LIKE** <pattern>;

**Example 1** : List the employees whose names start with the letter 'K'.

The SQL statement would be as follows:

**SELECT** emp_id,emp_name **FROM** Employee

**WHERE** emp_name LIKE 'K%';

**CHECK YOUR PROGRESS**

Q.2. Say TRUE or FALSE:

a) To display records which do not satisfy the conditions specified in the SQL expression, the NOT operator is used.

b) The NOT operator cannot be used in conjunction with any other operator in SQL.

c) While evaluating the SQL statements with multiple operators, the operators with higher precedence are evaluated first before evaluating those with lower precedence.

d) Logical operators has higher precedence than the arithmetic operators.

e) The LIKE, BETWEEN and IN operators are comparison operators.

f) The LIKE operator can include two "wildcard" characters underscore ( _ ) and percent sign (%).

Q.3. Fill in the blanks:

a) The NOT operator can also be used with a comparison operator to _____ the result of the comparison.

b) _____ determines the sequence in which the operations are performed in an SQL expression with multiple operators.

c) OR operator has _____ precedence than NOT.

d) LIKE searches for a specified _____ in a column of a database table.

e) The wildcard character underscore ( _ ) allows to match a _____ character.

f) The operators with higher precedence are evaluated _____ before evaluating those with lower precedence.

## 6.12 CONCATENATION OPERATOR

Concatenation is the process of combining strings. The Concatenation operator, ||, manipulates character strings. It appends one string to another. The result of concatenating two character strings is a character string itself. For example,

SELECT 'The employee name is' || emp_name

FROM Employee;

## 6.13 ALIAS COLUMN NAMES

When data from a database is selected, the heading of the column is same in the output as the column name in the database. The column heading can be changed in the output, that is, a different column heading can be given to a column in a table with the help of the alias column name of SQL. The alias column name can be assigned in the column list of the Select statement using the AS operator.

The syntax for alias column names is as given below:

**SELECT** column_name **AS** alias_name

**FROM** table_name

For Example: To display the names of the employees from the Employee table using alias column name, the following SQL statement is written.

**SELECT** emp_name **AS** Name

**FROM** Employee;

The output of the above statement would be as follows:

| **Name** |
|---|
| Kabita |
| Babita |
| Neel |
| Konika |
| Manoj |

## 6.14 ORDER BY CLAUSE

The ORDER BY clause is used to display the output table of a query in either ascending or descending alphabetical order. It sorts the individual rows of a table. *The syntax for the ORDER BY clause is:*

**SELECT** <*column_names*>

**FROM** <*Tablename*>

**WHERE** <*predicates*>

**ORDER BY** <*column_name*>*;*

The default sort order for ORDER BY clause is an ascending list, [a-z] for characters and [0-9] for numbers. SQL can also sort the records in descending order, [z-a]. The syntax is :

**SELECT** <*column_names*>

**FROM** <*Tablename*>

**WHERE** <*predicates*>

**ORDER BY** <*column_name*> *DESC;*

**Example 1:**

SELECT emp_name, salary

FROM Employee

ORDER BY salary ASC;

This would return the records sorted by the salary field in ascending order as given below:.

| emp_name | salary |
|---|---|
| Babita | 15000 |
| Konika | 16000 |
| Manoj | 17000 |
| Kabita | 20000 |
| Neel | 25000 |

**Example 2**:

SELECT emp_name, salary

FROM Employee

ORDER BY salary DESC;

This would return all the records sorted by the salary field in descending order as given below:

| emp_name | salary |
|---|---|
| Neel | 25000 |
| Kabita | 20000 |
| Manoj | 17000 |
| Konika | 16000 |
| Babita | 15000 |

## 6.15 HANDLING NULL VALUES

NULL value means unknown or missing data value. SQL treats any zero-length string like a NULL value. Sometimes there may be records in a table containing no value. This may be because during the data entry time the data was not available or for some rows in a table that particular field is not applicable. A table column, by default, can have NULL values. The NULL value is different from other values like a blank or a zero as zero is a numeric value and a blank space is a character value.

NULL values in a column of a database table can be tested by using the operators IS NULL and IS NOT NULL. Suppose to display the records with NULL values in the Phone column of the Employee table, the following expression can be written in SQL.

```
SELECT emp_name, phone FROM Employee
WHERE phone IS NULL;
```

Constraints can be applied to table columns to prevent the addition of invalid data or deletion of data that is required to maintain the overall consistency of the database. The constraints are used to control the data being entered into a database table.

In SQL, **NOT NULL** constraint can be defined at the column level in addition with the primary and foreign key. This constraint when defined on a column means that the column cannot be left empty. It becomes a mandatory column and a value must be entered into it.

The syntax for it is as:

```
<column_name> <datatype>(<size>) NOT NULL
```

For example, to create a student table, the SQL statement is –

```
CREATE TABLE Student(
Roll_no varchar2(10),
Name varchar2(20),
D_o_b date NOT NULL,
Address varchar2(30));
```

When inserting values in the columns of the Student table, the date of birth (d_o_b) field if not entered will display a error message. This field value has to be entered or each row in the table data.

The NOT NULL constraint can be applied only at the column level of a table.

## 6.16 DISTINCT CLAUSE

The SQL DISTINCT clause is used to eliminate the retrieving of the duplicate rows from a database table. It works with the SQL SELECT clause and it selects only the distinct or unique data from the database tables. For example, from the EMPLOYEE table of figure Table1, if we want to list the positions given to the employees, the SQL statement would be as :

SELECT **DISTINCT** position

FROM Employee;

The result of this query would be given as :

**Position**

Manager

Asst. Manager

AGM

Executive

The SQL DISTINCT expression returns a list of the position found in the 'position' column of the Employee table but it will remove the duplicates and will give only a single entry for each position.

### CHECK YOUR PROGRESS

Q.4. Fill in the blanks:

a) To combine strings, the _____ operator is used in SQL.

b) A column in a table can be given another name by using an _____ name.

c) The ORDER BY clause is used to display the records in either _____ or _____ alphabetical order.

d) Default sort order for ORDER BY clause is an _____ list.

e) Any zero-length string in SQL is treated as a _____.

f) NULL values can be tested by using the operators _____ and _____.

g) To prevent the addition of NULL values in a table, the _____ column constraint can be used.

h) The DISTINCT clause is used to retrieve the _____ records from a database table.

i) The concatenation operator used in SQL is _____.

j) The alias column name can be assigned in the column list of the Select statement using the _____ operator.

## 6.17 LET US SUM UP

- An *operator* is a symbol specifying an action that is performed on one or more expressions.
- *Compound conditions* are simple conditions that are joined by logical operators.
- AND, OR and NOT are logical operators.
- The ***AND*** operator in SQL joins together two or more conditions and returns result only when all of the conditions are true.
- The OR operator in SQL also connects two or more conditions but it returns results when any of the condition is true.
- The IN operator is used to compare a value to a list of values that has been specified and it returns TRUE if the compared value matches at least one of the values in the list.
- The BETWEEN operator is used with a WHERE clause to test whether a value lies in a specified range of values.
- The NOT operator is used to display only those records which do not satisfy the given conditions.
- Operator precedence determines the sequence in which the operations are performed in an SQL expression with multiple operators.

- While evaluating the SQL statements with multiple operators, the operators with higher precedence are evaluated first before evaluating those with lower precedence.
- The LIKE operator is used with a WHERE clause to compare a character string to a specified pattern.
- The LIKE operator uses two “wildcard” characters underscore ( _ ) and percent sign ( % ).
- The Concatenation operator, ||, manipulates character strings and appends one string to another.
- A column in a table can be given another name in the output set by using an alias column name.
- The ORDER BY clause is used to display the output table of a query in either ascending or descending alphabetical order. It sorts the individual rows of a table.
- SQL treats any zero-length string like a NULL value.
- The NULL value is different from other values like a blank or a zero as zero is a numeric value and a blank space is a character value.
- NULL values can be tested by using the operators IS NULL and IS NOT NULL.
- NOT NULL constraint can be defined at the column level to prevent the addition of NULL value in a column to maintain the overall consistency of the database.
- DISTINCT clause is used to display the unique rows from a database table.

## 6.18 FURTHER READINGS


- *Sams Teach Yourself SQL in 24 Hours*, Ryan Stephens, Ron Plew, Arie Jones, Sams Publishing.
- *SQL for Dummies*, *Allen G. Taylor*, Wiley Publishing Inc.
- *SQL, PL/SQL The Programming Language Of Oracle 4th Revised Edition*, Bayross, Ivan, Bpb publications.

## 6.19 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1 :** a) Operator, b) Compound conditions, c) True, d) any, e) IN, f) TRUE

**Ans. to Q. No. 2 :** a) True, b) False, c) True, d) False, e) True, f) True

**Ans. to Q. No. 3 :** a) negate, b) Operator precedence, c) lower d) pattern, e) single, f) first

**Ans. to Q. No. 4 :** a) Concatenation, b) alias, c) Ascending, descending, d) Ascending, e) NULL value, f) IS NULL, IS NOT NULL, g) NOT NULL, h) Unique, i) ||, j) AS

## 6.20 MODEL QUESTIONS

Q.1. Explain the SQL logical operators with examples.

Q.2. What is the difference between LIKE operator and BETWEEN operator?

Q.3. How can you sort the records of a table in the output table?

Q.4. What do you mean by NULL value? How can be a NULL value in any column be tested? How can you prevent NULL values in a column of a table?

# UNIT 7 : RELATIONAL DATABASE DESIGN

## UNIT STRUCTURE



## 7.1 LEARNING OBJECTIVES

After going through this unit, you will able to:

- define normalization
- define anomalies of unnormalized database
- learn about First normal form
- learn about Second normal form
- learn about Third normal form

## 7.2 INTRODUCTION

The goal of database normalization is to decompose relations with anomalies in order to produce smaller, well-structured relations. Normalization usually involves dividing large tables into smaller (and less redundant) tables and defining relationships between them. The objective is to isolate data so that additions, deletions, and modifications of a field can be made in just one table and then propagated through the rest of the database via the defined relationships.

Edgar F. Codd, the inventor of the relational model, introduced the concept of normalization and what we now know as the First Normal Form (1NF) in 1969.Codd went on to define the Second Normal Form (2NF) and Third Normal Form (3NF) in 1971, and Codd and Raymond F. Boyce defined the Boyce-Codd Normal Form (BCNF) in 1974.

A standard piece of database design guidance is that the designer should create a fully normalized design; selective denormalization can subsequently be performed for performance reasons. However, some modeling disciplines, such as the dimensional modeling approach to data warehouse design, explicitly recommend non-normalized designs, i.e. designs that in large part do not adhere to 3NF.

## 7.3 NORMALIZATION

***Normalization*** is the process of organizing data in a database. This includes creating tables and establishing relationships between those tables according to rules designed both to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.

Redundant data wastes disk space and creates maintenance problems. If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations. A customer address change is much easier to implement if that data is stored only in the Customers table and nowhere else in the database.

There are a few rules for database normalization. Each rule is called a "***normal form***." If the first rule is observed, the database is said to be in "first normal form." If the first three rules are observed, the database is considered to be in "third normal form." Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.

As with many formal rules and specifications, real world scenarios do not always allow for perfect compliance. In general, normalization requires additional tables and some customers find this cumbersome. If you decide

to violate one of the first three rules of normalization, make sure that your application anticipates any problems that could occur, such as redundant data and inconsistent dependencies.

**Objectives of normalization**

1. To free the collection of relations from undesirable insertion, update and deletion dependencies;
2. To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs;
3. To make the relational model more informative to users;
4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

## 7.4 ANOMALIES OF UNNORMALIZED DATABASE

**Database anomalies :** When an attempt is made to modify (update, insert into, or delete from) a table, undesired side-effects may follow. Not all tables can suffer from these side-effects; rather, the side-effects can only arise in tables that have not been sufficiently normalized. An insufficiently normalized table might have one or more of the following characteristics:

**Update anomaly :** The same information can be expressed on multiple rows; therefore updates to the table may result in logical inconsistencies. For example, each record in an "Employees' Skills" table might contain an Employee ID, Employee Address, and Skill; thus a change of address for a particular employee will potentially need to be applied to multiple records (one for each of his skills). If the update is not carried through successfully—if, that is, the employee's address is updated on some records but not others—then the table is left in an inconsistent state. Specifically, the table provides conflicting answers to the question of what this particular employee's address is. This phenomenon is known as an update anomaly.

**Insertion anomaly :** There are circumstances in which certain facts cannot be recorded at all. For example, each record in a "Faculty and Their Courses" table might contain a Faculty ID, Faculty Name, Faculty Hire Date,

and Course Code—thus we can record the details of any faculty member who teaches at least one course, but we cannot record the details of a newly-hired faculty member who has not yet been assigned to teach any courses except by setting the Course Code to null. This phenomenon is known as an insertion anomaly.

**Deletion anomaly :** There are circumstances in which the deletion of data representing certain facts necessitates the deletion of data representing completely different facts. The "Faculty and Their Courses" table described in the previous example suffers from this type of anomaly, for if a faculty member temporarily ceases to be assigned to any courses, we must delete the last of the records on which that faculty member appears, effectively also deleting the faculty member. This phenomenon is known as a deletion anomaly.

**Functional Dependency :** A functional dependency is an important part of designing databases in the relational model, and in database normalization and denormalization. The functional dependencies, along with the attribute domains, are selected so as to generate constraints that would exclude as much data inappropriate to the user domain from the system as possible.

A dependency occurs in a database when information stored in the same database table uniquely determines other information stored in the same table. You can also describe this as a relationship where knowing the value of one attribute (or a set of attributes) is enough to tell you the value of another attribute (or set of attributes) in the same table.

Saying that there is a dependency between attributes in a table is the same as saying that there is a functional dependency between those attributes. If there is a dependency in a database such that attribute B is dependent upon attribute A, you would write this as A -> B.

This example illustrates the concept of functional dependency. The situation modelled is that of college students visiting one or more lectures in each of which they are assigned a teaching assistant (TA). Let’s further assume that every student is in some semester and is identified by a unique integer ID.

| StudentID | Semester | Lecture | TA |
|---|---|---|---|
| 1234 | 8 | Compiler | Ramakant |
| 2380 | 4 | Compiler | Pramil |
| 1234 | 8 | FLA | Amin |
| 1201 | 4 | Compiler | Pramil |
| 1201 | 4 | Physics II | Simone |

We notice that whenever two rows in this table feature the same StudentID, they also necessarily have the same Semester values. This basic fact can be expressed by a functional dependency:

- StudentID -->Semester.

Other nontrivial functional dependencies can be identified, for example:

- {StudentID, Lecture} -->TA
- {StudentID, Lecture} -->{TA, Semester}

The latter expresses the fact that the set {StudentID, Lecture} is a superkey of the relation.

**Trivial Functional Dependencies:** A **trivial functional dependency** occurs when you describe a functional dependency of an attribute on a collection of attributes that includes the original attribute.

For example, "{A, B} --> B" is a trivial functional dependency, as is {name, SSN} --> SSN. This type of functional dependency is called trivial because it can be derived from common sense. It is obvious that if you already know the value of B, then the value of B can be uniquely determined by that knowledge.

A **full functional dependency** occurs when you already meet the requirements for a functional dependency and the set of attributes on the left side of the functional dependency statement cannot be reduced any farther. For example, in a table listing employee characteristics including Social Security Number (SSN) and name.{SSN, age} -> name is a functional dependency, but it is not a full functional dependency because you can remove age from the left side of the statement without impacting the dependency relationship.

**Properties of functional dependencies :** Given that X, Y, and Z are sets of attributes in a relation R, one can derive several properties of functional dependencies. Among the most important are Armstrong's axioms, which are used in database normalization:

- **Subset Property** (Axiom of Reflexivity): If Y is a subset of X, then X -->Y
- **Augmentation** (Axiom of Augmentation): If X -->Y, then XZ -->YZ
- **Transitivity** (Axiom of Transitivity): If X -->Y and Y --> Z, then X --> Z

From these rules, we can derive these secondary rules:

- **Union**: If X --> Y and X -->Z, then X --> YZ
- **Decomposition**: If X --> YZ, then X -->Y and X --> Z

**Pseudotransitivity**: If X -->Y and WY --> Z, then WX --> Z

Equivalent sets of functional dependencies are called covers of each other. Every set of functional dependencies has a canonical cover.

**Multivalued dependencies** occur when the presence of one or more rows in a table implies the presence of one or more other rows in that same table. For example, imagine a car company that manufactures many models of car, but always makes both red and blue colors of each model. If you have a table that contains the model name, color and year of each car the company manufactures, there is a multivalued dependency in that table. If there is a row for a certain model name and year in blue, there must also be a similar row corresponding to the red version of that same car.

**Importance of Dependencies :** Database dependencies are important to understand because they provide the basic building blocks used in database normalization. For example:

- For a table to be in second normal form (2NF), there must be no case of a non-prime attribute in the table that is functionally dependendent upon a subset of a candidate key.
- For a table to be in third normal form (3NF), every non-prime attribute must have a non-transitive functional dependency on every candidate key.

- For a table to be in Boyce-Codd Normal Form (BCNF), every functional dependency (other than trivial dependencies) must be on a superkey.

**CHECK YOUR PROGRESS**

Q.1. Select the correct answer :

i) A functional dependency in which one or more nonkey attributes are functionally dependent on part but not all of the primary key is called a _____ dependency.

A) Partial key-based B) Partial functional

C) Cross key D) Merged relation

ii) The different classes of relations created by the technique for preventing modification anomalies are called:

A) normal forms

B) referential integrity constraints

C) functional dependencies

D) None of the above is correct.

iii) Which of the following are anomalies that can be caused by redundancies in tables?

A) Insertion anomaly B) Deletion anomaly

C) Modification anomaly D) all of the above

iv) A(n) _____ is a constraint between two attributes.

A) Action Assertion B) Functional Dependency

C) Candidate D)Determinant

v) A _____ is a dependency of one non primary key attribute on another non primary key attribute.

A) Functional dependency

B) Partial dependency

C) Transitive dependency

D) Non primary key dependency.

## 7.5 NORMAL FORM : 1NF

The first normal form (1NF) sets the very crucial rule to create the database :

1. There are no repeating or duplicate fields.
2. Each cell contains only a single value.
3. Each record is unique and identified by primary key.

The normalization process involves getting our data to adjust in a progressive normal forms and you cannot achieve the higher level of normalization without satisfying the previous levels. The First Normal Form asking the values in each cell of a table must be atomic. The word atomic describes that there should be no sets of values in one particular cell.

Violation of any of these conditions would mean that the table is not strictly relational, and therefore that it is not in 1NF.

**Example 1 :** Lets explore this principle with an example – a table within a human resources database that stores the manager-subordinate relationship. For the purposes of our example, we will impose the business rule that each manager may have one or more subordinates while each subordinate may have only one manager.

Intuitively, when creating a table to track this information, we might create a table with the following fields:

- Manager
- Subordinate1
- Subordinate2
- Subordinate3
- Subordinate4

However, recall the first rule imposed by 1NF: eliminate duplicative columns from the same table. Clearly, the Subordinate1-Subordinate4 columns are duplicative. Take a moment and ponder the problems raised by this scenario. If a manager only has one subordinate – the Subordinate2-Subordinate4 columns are simply wasted storage space (a precious database commodity). Furthermore, imagine the case where a manager

already has 4 subordinates – what happens if she takes on another employee? The whole table structure would require modification.

At this point, a second bright idea usually occurs to database novices: We don't want to have more than one column and we want to allow for a flexible amount of data storage. Let's try something like this:

- Manager
- Subordinates

where the Subordinates field contains multiple entries in the form "Mary, Bill, Joe"

This solution is closer, but it also falls short of the mark. The subordinates column is still duplicative and non-atomic. What happens when we need to add or remove a subordinate? We need to read and write the entire contents of the table. That's not a big deal in this situation, but what if one manager had one hundred employees? Also, it complicates the process of selecting data from the database in future queries.

Here is a table that satisfies the first rule of 1NF:

- Manager
- Subordinate

In this case, each subordinate has a single entry, but managers may have multiple entries.

Now, what about the second rule: identify each row with a unique column or set of columns (the primary key)? You might take a look at the table above and suggest the use of the subordinate column as a primary key. In fact, the subordinate column is a good candidate for a primary key due to the fact that our business rules specified that each subordinate may have only one manager. However, the data that we have chosen to store in our table makes this a less than ideal solution. What happens if we hire another employee named Jim? How do we store his manager-subordinate relationship in the database?

It is best to use a truly unique identifier (such as an employee ID) as a primary key. Our final table would look like this:

- Manager ID
- Subordinate ID

**Example 2 :** UN-normalized table: Students

| Student_id | Advisor | Adv-Room | Class1 | Class2 | Class3 |
|---|---|---|---|---|---|
| 1022 | Jones | H412 | 101-2 | 112-01 | 155-01 |

First Normal Form: No repeating groups

Tables should have only two dimensions. Since one student has several classes, these classes should be listed in a separate

table. Fields Class1, Class2, & Class3 in the above record are indications of design trouble.

## 7.6 NORMAL FORM : 2NF

Second Normal Form (2NF): A 1NF table is in 2NF if and only if, given any candidate key K and any attribute A that is not a constituent of a candidate key, A depends upon the whole of K rather than just a part of it.

2NF attempts to reduce the amount of redundant data in a table by extracting it, placing it in new tables and creating relationships between those tables.

**Example :** Conside a table called Customers with the following elements:

- CustNum
- FirstName
- LastName
- Address
- City
- State
- ZIP

A brief look at this table reveals a small amount of redundant data. We're storing for state zip and city. Now, that might not seem like too much added storage in our simple example, but if we had thousands of rows in our table. Additionally, if the ZIP code for city were to change, we'd need to make that change in many places throughout the database.

In a 2NF-compliant database structure, this redundant information is extracted and stored in a separate table. Our new table (let's call it ZIPs) might have the following fields:

- ZIP
- City
- State

Now that we've removed the duplicative data from the Customers table, we've satisfied the first rule of second normal form. We still need to use a foreign key to tie the two tables together. We'll use the ZIP code (the primary key from the ZIPs table) to create that relationship. Here's our new Customers table:

- CustNum
- FirstName
- LastName
- Address
- ZIP

We've now minimized the amount of redundant information stored within the database and our structure is in second normal form!

**Example 2 :**

Un normalized Student table

| Student# | AdvID | AdvName | AdvRoom | Class1 | Class2 |
|---|---|---|---|---|---|
| 123 | 123A | James | 555 | 102-8 | 104-9 |
| 124 | 123B | Smith | 467 | 209-0 | 102-8 |

Student Table in 1NF

| Student# | AdvID | AdvName | AdvRoom | Class# |
|---|---|---|---|---|
| 123 | 123A | James | 555 | 102-8 |
| 123 | 123A | James | 555 | 104-9 |
| 124 | 123B | Smith | 467 | 209-0 |
| 124 | 123B | Smith | 467 | 102-8 |

Student Table in 2NF

| Student# | AdvID | AdvName | AdvRoom |
|---|---|---|---|
| 123 | 123A | James | 555 |
| 124 | 123B | Smith | 467 |

| Student# | Class# |
|---|---|
| 123 | 102-8 |
| 123 | 104-9 |

| | |
|---|---|
| 124 | 209-0 |
| 124 | 102-8 |

## 7.7 NORMAL FORM : 3NF

Third Normal Form (3NF): There are two basic requirements for a database to be in third normal form:

- Already meet the requirements of both 1NF and 2NF
- Remove columns that are not fully dependent upon the primary key.

**Example :** An example of a 2NF table that fails to meet the requirements of 3NF is:

Tournament Winners

| Tournament | Year | Winner | Winner Date of Birth |
|---|---|---|---|
| Indian Invitational | 1998 | Al Fredrickson | 21 July 1975 |
| US Open | 1999 | Bob Albertson | 28 Sep 1968 |
| Grand Masters | 1999 | Al Fredrickson | 21 July 1975 |
| Indiana Invitational | 1999 | Chip Masterson | 14 Mar 1977 |

Because each row in the table needs to tell us who won a particular Tournament in a particular Year, the composite key {Tournament, Year} is a minimal set of attributes guaranteed to uniquely identify a row. That is, {Tournament, Year} is a candidate key for the table.

The breach of 3NF occurs because the non-prime attribute Winner Date of Birth is transitively dependent on the candidate key {Tournament, Year} via the non-prime attribute Winner. The fact that Winner Date of Birth is functionally dependent on Winner makes the table vulnerable to logical inconsistencies, as there is nothing to stop the same person from being shown with different dates of birth on different records.

In order to express the same facts without violating 3NF, it is necessary to split the table into two:

Tournament Winners

| Tournament | Year | Winner |
|---|---|---|
| Indian Invitational | 1998 | Al Fredrickson |

| | | |
|---|---|---|
| US Open | 1999 | Bob Albertson |
| Grand Masters | 1999 | Al Fredrickson |
| Indian Invitational | 1999 | Chip Masterson |

Player Dates of Birth

| Player | Date of Birth |
|---|---|
| Chip Masterson | 14 March 1977 |
| Al Fredrickson | 21 July 1975 |
| Bob Albertson | 28 Sep 1968 |

Update anomalies cannot occur in these tables, which are both in 3NF.

**Example :**

**Students Table**

| Student# | AdvID | AdvName | AdvRoom |
|---|---|---|---|
| 123 | 123A | James | 555 |
| 124 | 123B | Smith | 467 |

**Student Table**

| Student# | AdvID |
|---|---|
| 123 | 123A |
| 124 | 123B |

**Advisor Table**

| AdvID | AdvName | AdvRoom |
|---|---|---|
| 123A | James | 555 |
| 123B | Smith | 467 |

**Tables in 3NF**

**Students Table**

| Student# | AdvID |
|---|---|
| 123 | 123A |
| 124 | 123B |

**Registration Table**

| Student# | Class# |
|---|---|
| 123 | 102-8 |
| 123 | 104-9 |

| | |
|---|---|
| 124 | 209-0 |
| 124 | 102-8 |

**Advisor Table**

| AdvID | AdvName | AdvRoom |
|---|---|---|
| 123A | James | 555 |

**CHECK YOUR PROGRESS**

Q.2. Select the correct answer :

i) A relation that contains no multivalued attributes, and has nonkey attributes solely dependent on the primary key, but contains transitive dependencies is in which normal form?

A) First B) Second

C) Third D) Fourth

ii) When the value of every field in a table is dependent on the primary key of that table and nothing else, that table is said to be in:

A) second normal form

B) a one-to-many relationship

C) third normal form D) good normal form.

iii) The 2NF describes the tabular format in which

A) There are no repeating groups in the table

B) All attributes are dependent on primary key

C) A & B with no partial dependency

D) there is no partial dependency

iv) Normalization _____ data duplication.

A) eliminates B) reduces

C) increases D) maximizes

v) In _____ normal form, any transitive dependencies

have been removed.

A) First B) Second

C) Third D) Fourth

vi) A table is in second normal form (2NF) if:

A) It includes no partial dependencies

B) It includes no transitive dependencies

C) It is in first normal form (1NF) and it includes no partial dependencies

D) It is in first normal form (1NF) and it includes no transitive dependencies

## 7.8 LET US SUM UP

- The process of organizing data to minimize redundancy is called normalization.
- The goal of database normalization is to decompose relations with anomalies in order to produce smaller, well-structured relations.
- Database anomalies **-**When an attempt is made to modify (update, insert into, or delete from) a table, undesired side-effects may follow.
- A table is in 1NF if and only if there's no top-to-bottom ordering to the rows,there's no left-to-right ordering to the columns,there are no duplicate rows,every row-and-column intersection contains exactly one value from the applicable domain,all columns are regular i.e. rows have no hidden components such as row IDs, object IDs.
- A 1NF table is in 2NF if and only if, given any candidate key K and any attribute A that is not a constituent of a candidate key, A depends upon the whole of K rather than just a part of it.
- A table is in 3NF if it already meet the requirements of both 1NF and 2NF and remove columns that are not fully dependent upon the primary key.

## 7.9 ANSWERS TO CHECK YOUR PROGRESS

**Ans. to Q. No. 1 :** i) B, ii) A, iii) D, iv) B, v) C.

**Ans. to Q. No. 2 :** i) B, ii) A, iii) C, iv) A, v) C, vi) C

## 7.10 FURTHER READINGS

- *Database System Concepts* By Henry korth and A.Silberschatz.
- *An Introduction to Database System* by Bipin Desai.
- *File Structure* by Michael J. Folk, Greg, Riccardi.

## 7.11 MODEL QUESTIONS

Q.1. Show that every 3NF schema is also in 2NF.

Q.2. Given the three goals of relational database design, is ther any reason to design a database schema that is in 2NF, but is in no higher order normal form?

Q.3. Why are some functional dependencies called trivial?

Q.4. Give a set of FDs for the relation schema R(A,B,C,D) with primary key AB under which R is in 1NF but not in 2NF.

Q.5. Give a set of FDs for the relation schema R(A,B,C,D) with primary key AB under which R is in 2NF but not in 3NF.

Q.6. Discuss the differnt database anomalies.

Q.7. Design a database for a bank with normalization applied on each of the tables.