Series

3rd Edition

A Fundamental Study of

# DATABASE MANAGEMENT SYSTEMS

- More than 275 Short Questions
- More Than 300 Multiple Choices
- More than 275 True/False
- Complete SQL & Access Covered
- MS Access Project Included

Imran Saeed Tasleem Mustafa
Tariq Mahmood Ahsan Raza Sattar

IT Series

A Fundamental Study of

# DATABASE

# MANAGEMENT SYSTEMS

(3rd Edition)

*Written by*


**Imran Saeed**
MSc (CS), JCP, MCP

**Tasleem Mustafa**
MS (CS), M.Sc. (CS), M.Sc. (Math), MIS (USA), OCP

**Tariq Mahmood**
MSc (CS), JCP, MCP

**Ahsan Raza Sattar**
MS (CS), MSc (CS), MBA, OCP



*Published by:*
IT Series Publications

**Published by:** IT Series Publications
Mob: (+92) 0321 661 56 56

**Price:** **Rs. 285/-**

***Can be had from:***

Kitab Markaz
**Aminpur Bazar, Faisalabad, Pakistan**
**Ph: +92-41-2642707**

Imtiaz Book Depot
**Urdu Bazar, Lahore, Pakistan.**
**Ph: +92-42-7235944**

# Preface

The basic objective for writing this book is to provide complete guidance for the students to learn the basic concepts of database management systems. To fulfill this purpose, we have included hundreds of practical examples for the students.

The book includes, besides theoretical portion, complete chapters on **SQL** and **MS Access** so that the readers may get a practical knowledge of databases. These chapters contain many practical examples that make it simple and easy for readers tc implement the topics they learn.

To cover the requirements of the students with respect to examination, there are hundreds of **short questions**, **multiple choice** and **true/false** questions in the book so that the students may prepare for the examination from different point of views.

Though, the book is basically meant for the students, but it can also be used by other people who want to get the basic knowledge of databases.

Readers are welcome to send suggestions for improvements of the book by sending email to us at **comments@itseries.com.pk**. You can also visit our website **www.itseries.com.pk** for any interaction and latest information about the book.

***Authors***

## Table of Contents

---

CHAPTER 1

# Introduction to Databases

## Chapter Overview

## 1.1 Data

A collection of raw facts and figures is called **data**. The word **raw** means that the facts have not yet been processed to get their exact meaning. Data is collected from different sources. It is collected for different purposes. Data may consist of numbers, characters, symbols or pictures etc.

Data is a vital resource for any organization. A **resource** is anything that is valuable for an organization. Resources inlclude buildings, furniture, vehicles, machinery and employees. Data is considered a resource because it provides correct information for making proper and timely decisions. It enables the management of the organization to utilize other resources effectively. It is not possible to make good decision if data is not available in desired format.

### 1.1.1 Examples of Data

1. When students get admission in colleges or universities, they have to fill out an admission form. The form consists of raw facts about the students. These raw facts are student's name, father name, address etc. The purpose of collecting data is to maintain the records of students during their study period in college or university.
2. During census, Government of Pakistan collects the data of all citizens. Government stores this data permanently to use it for different purposes at different times.
3. Different organizations conduct surveys to know the opinion of the people about their product. In these surveys, people express their ideas and opinions about different issues. These ideas and opinions of the people are stored as data. The organizations use this data for the improvement of their products.

### 1.1.2 Types of Data

Data may be of the following types:

#### 1. Numeric Data

Numeric data consists of numeric digits from 0 to 9 like 10, 245 or –5. The numeric type of data may either be positive or negative.

#### 2. Alphabetic Data

Alphabetic data consists of alphabetic letters from A to Z, a to z and blank spaces e.g. "IT Series", "Computer" and "Islam" etc.

#### 3. Alphanumeric Data

Alphanumeric data consists of numeric digits (0 to 9), letters (A to Z) and all special characters like +, %, and @ etc. like "87%", "$300" and "H#17".

#### 4. Image Data

This type of data includes charts, graphs, pictures and drawings. This form of data is more comprehensive. It can be transmitted as a set of bits.

#### 5. Audio Data

Sound is a representation of audio. Audio data includes music, speech or any type of sound.

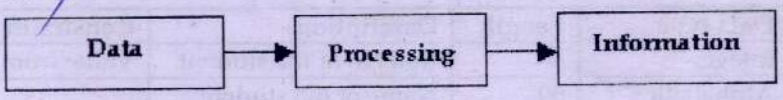
#### 6. Video Data

Video is a set of full-motion images played at a high speed. Video is used to display actions and movements.

## 1.2 Information

The processed data is called **information**. Information is an organized and processed form of data. It is more meaningful than data and is used for making decisions. Data is used as input for the processing and information is the output of this processing. This information can be used again in some other processing and will be considered as data in that processing.

For example, the marks of a student in different subjects is the data. To calculate the total marks, the marks of different subjects are used as data and total marks is the information. Now, to calculate the average marks of the student, this information will be processed again. In this processing, the information is used as data and average marks will be the information.




### 1.2.1 Examples of Information

Some examples of information are as follows:

1. In colleges and universities, the raw facts about students are stored on admission forms. If we want to find out a list of all students who live in Faisalabad, we will apply some processing on this data. This processing will give us the desired list. This list is a form of processed data and will be called information.
2. The data stored in census is used to generate different type of information. For example, government can use it to find total number of graduates or literacy rate in country etc. This information can be obtained by processing stored data. Government can use this information to take important decisions to improve literacy rate.
3. An organization can use the opinion of the people as data and process it to generate information of its interest. For example, it can know that how many people of the country are satisfied with the quality of its product and how many are unsatisfied. The organization can use this information for the improvement of its product.

## 1.3 Difference between Data and Information

The difference between data and information as follows:

| Data | Information |
|---|---|
| 1. Data consists of unprocessed raw facts. | 1. Information is the processed form of data. |
| 2. Data is used as input in the computer. | 2. Information is the output of computer. |
| 3. Data is not meaningful. | 3. Information is meaningful. |
| 4. Data is normally huge in its volume. | 4. Information is normally short in volume. |
| 5. Data is the asset of organizations and is not available to people for sale. | 5. Information is normally available to people for sale. |
| 6. Data is difficult or even impossible to reproduce. Suppose Government loses data of census. It is almost impossible to reproduce it. | 6. Information is easier to reproduce if lost. For example, if the list of illiterate citizens is lost, it can be reproduced easily from stored data. |
| 7. Data is used rarely. | 7. Information is used frequently. |
| 8. Data is an independent entity. | 8. Information depends on data. |
| 9. Data is not used in decision-making. | 9. Information is very important for decision-making. |

## 1.4 Metadata

Metadata can be defined as data about data. It is used to describe the properties and characteristics of some other data. Metadata describes the size, format and other characteristics of data. It also includes the rules and constraints about data.

### Example

When you create a table, you specify the data type, size, format and other constraints for entering data in different fields of the table. This is metadata of the table. It describes the properties of the data to be stored in the table. Metadata is very important to ensure the integrity of the data.

| Field name | Data type | Length | Description | Constraint |
|---|---|---|---|---|
| Roll No | Integer | 3 | Roll No of the student | Value from 1 to 100 |
| Name | Alphabetic | 50 | Name of the student | |
| Address | Alphanumeric | 100 | Address of the student | |
| Email | Alphanumeric | 25 | Email of the student | Must contain @ and . |
| Phone | Alphanumeric | 25 | Phone of the student | |

**Table:** Example of metadata

| Roll no | Name | Address | Email | Phone |
|---|---|---|---|---|
| 1 | Ejaz | Faisalabad | ejaz@itseries.com.pk | 784682 |
| 2 | Usman | Faisalabad | usman@itseries.com.pk | 727253 |

**Table:** Example of a database table

## 1.5 File Processing System

Traditional or simple file processing is the first computer-based method to handle business application. In the past, many organizations stored data in files on tape or disk. The data was managed using file-processing system. In a typical file processing system, each department in an organization has its own set of files. The files are designed specially for their own applications. The records in one file are not related to the records in any other file.

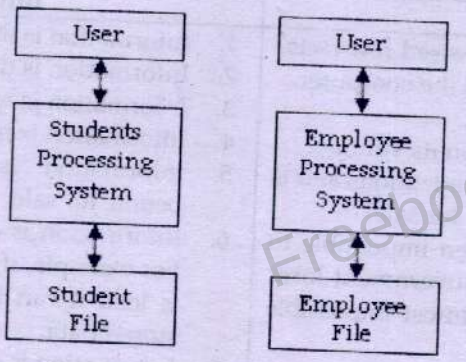



**Figure:** File Processing System

Business organizations have used file-processing system for many years. But this system has many disadvantages.

## 1.5.1 Disadvantages of File Processing System

Some important disadvantages of file processing system are as follows:

### 1. Data Redundancy and Inconsistency

In file processing system, the same data may be duplicated in several files. Suppose there are two files "Students" and "Library". The file "Students" contains the Roll No, name, address and telephone number and other details of all students in a college. The file "Library" contains the Roll No and name of those students who get a book from library along with the information about the rented books. The data of one student appears in two files. This is known as data redundancy. This redundancy causes higher storage.

This situation can also result in data inconsistency. Inconsistency means that two files may contain different data of the same student. For example, if the address of a student is changed, it must be changed in both files. There is a possibility that it is changed in the "Students" file and not from "Library" file. The data becomes inconsistent in this situation.

### 2. Data Isolation

The data in file processing system is stored in various files. It becomes very difficult to write new application programs to retrieve the appropriate data. Suppose that student emails are stored in "Students" file and fee information is stored in "Fee" file. The data from both files is required to send an email message to inform a student that the date for fee payment is over. In file processing system, it is difficult to generate such type of list from multiple files.

### 3. Integrity Problems

Integrity means reliability and accuracy of data. The stored data must satisfy certain types of consistency constrains. For example, Roll No and Marks of students should be numeric value. It is very difficult to apply these constrains on files in file processing system.

### 4. Program Data Dependency

Program data dependency is a relationship between data in files and program required to update and maintain the files. Application programs are developed according to a particular file format in file processing system. If the format of file is changed, the application program also needs to be changed accordingly. For example, if there is a change in the length of postal code, it requires change in the program. The changes may be costly to implement.

### 5. Atomicity Problem

An operation on data may consist of different steps. A collection of all steps required to complete a process is known as **transaction**. The atomicity means that either one transaction should take place as a whole or it should not take place at all. Suppose a user wants to transfer money from account A to account B. This process consists of two steps:

1. Deduct the money from account A.
2. Add the money to account B.

Suppose that the system fails when the computer has performed the first step. It means that the amount has been deducted from account A but has not been added to account B. This situation can make data inconsistent. File processing system does not provide the facility to ensure atomicity of data.

### 6. Security Problems

File processing system does not provide adequate security on data. In some situations, it is required to provide different types of access to data for different users. For example, a

data entry operator should only be allowed to enter data. The chairman of the organization should be able to access or delete the data completely. Such types of security options are not available in file processing system.

### 7. Program Maintenance

The programs developed in file processing system are difficult to maintain. Most of the budget may be spent on maintenance. It makes it difficult to develop new applications.

## 1.6 Database

Database is an **organized** collection of **related** data that is stored in an **efficient** and **compact** manner. The word **organized** means that data is stored in such a way that the user can use this data easily. The word **related** means that a database is normally created to store the data about a particular topic.

For example if a database is created for students, it will contain data about the students such as roll no, name, address etc. Similarly, if the database is about the employees of an organization, it will contain the data of employees such as employee ID, grade and salary etc. All data in database is arranged in tables.

The word **efficient** means that the user can search the required data quickly. The word **compact** means that the stored data occupies as little space as possible in computer.

### Tables

Table is the fundamental object of the database structure. The basic purpose of a table is to store data. A table consists of rows and columns. A table is a very convenient way to store data. The data in tables can be manipulated easily.

| Serial No | Name | Qualification | Email |
|---|---|---|---|
| 1 | Usman | B.Sc. | usman@itseries.com.pk |
| 2 | Abdullah | M.Sc. | abdullah@itseries.com.pk |
| 3 | Ejaz | M.Sc. | ejaz@itseries.com.pk |

**Table:** Records table

### Rows / Record

Rows are the horizontal part of the table. It is a collection of related fields. For example, the above table has three rows. Each row contains a record of different person.

| | | | |
|---|---|---|---|
| 2 | Usman | B.Sc. | usman@itseries.com.pk |

A single row/record

### Columns / Field

Columns are the vertical part of the table. For example, all values in the above table under "Name" field make a column.

| Name |
|---|
| Usman |
| Abdullah |
| Ejaz |

A single column

### 1.6.1 Examples of Databases

Following are some important examples of databases:

#### 1. Phone Directory

Phone directory is a simple example of a database. A phone directory stores the phone numbers of different persons. Searching a phone number from phone directory is very easy as all phone numbers are stored in an organized way.

#### 2. Library

A library contains thousands of books. It is very difficult to handle the records of all these books without database. A database system can be used to store the records of books, members of the library, issuance and recovery of books etc. The database can be used to search the required books easily. This database can help for doing research work.

#### 3. Accounts

A database is used to control the accounts system of an organization. The accounts database keeps the record of all financial transactions of the organization. It can be used to perform different calculations to find information about business such as annual profit, trial balance and ledger etc.

#### 4. College

A college has many students in different classes. A database may be used to keep the records of the students, fee transactions, examination information and other data of the college. It can also store the attendance of the students.

## 1.7 Database Management System

A **database management system (DBMS)** is a collection of programs that are used to create and maintain a database. DBMS is a general-purpose software system that provides the following facilities:

1. It provides the facility to define the structure of database. The user can specify data types, format and constraints for the data to be stored in database.
2. It provides the facility to store data on some storage medium controlled by DBMS.
3. It provides the facilities to insert, delete, update and retrieve specific data to generate reports etc.

## 1.8 Components of Database Environment

The important components of a database environment are as follows:

#### 1. Repository

A repository is a collection of all data definitions, data relationships, output styles and report formats etc. All this information is the metadata that is important to manage database.

#### 2. Database Management System

A database management system (DBMS) is a collection of programs that are used to create and maintain a database.

#### 3. Database

Database can be defined as an organized collection of related data. The word "organized" means that data is stored in such a way that the user can store, manipulate and

retrieve data easily. The word "related" means that a database is normally created to store the data about a particular topic.

### 4. Application Program

An application program is a program that is used to send commands to the database management system to manipulate database. These commands are sent to the DBMS through graphical user interface. The user interacts with the application program and the application program further interacts with the database management system. Two important application programs are Developer2000 and Power Builder.

### 5. User Interface

The user interface is a visual environment that is used by the user to communicate with the computer. It consists of menus, buttons and other components. All windows based software use graphical user interface. The user interface consists of following components:

- **Forms:** The forms are used to enter data in the database. A form consists of textboxes, labels and buttons that are used by the users for entering data easily. The user can also retrieve, change and update data by using forms.
- **Menus:** Menus are a list of commands for performing different operations. Menus are frequently used in windows-based applications. The user can use them easily for manipulating the database.
- **Reports:** Reports are the output of the database application. The user can generate different types of reports by manipulating the database. The information on the reports is arranged in different forms and may contain graphs, charts and tables etc.

### 6. Data Administrators

Data administrators are the persons who are responsible of whole information system. They authorize access to the database as well as coordinate and monitor the use of database.

### 7. System Analysts and Application Programmers

System analysts determine the requirements of end users and develop specifications for transactions. Application programmers implement these specifications and programs.

### 8. End User

End users are those persons who interact with the application directly. They are responsible to insert, delete and update data in the database. They get information from the system as and when required.

## 1.9 Database Approach

Database approach has many advantages over file processing system.

### 1.9.1 Advantages of Database Approach

Some important advantages of database approach are as follows:

### 1. Redundancy Control

The data in a database appears only once and is not duplicated. For example the data of a student in college database is stored in one table. The table can be accessed for different purposes. For example, if we want to store the marks of the student in a table, only Roll No of the student will be used in second table. The second table will be connected to the student table for accessing the information about the student as follows:

**Student Table**

| Roll No | Name | Address | Email | Phone |
|---|---|---|---|---|
| 1 | Usman | Faisalabad | usman@itseries.com.pk | 727253 |
| 2 | Abdullah | Toronto | Abdullah@itseries.com.pk | 784692 |

**Marks Table**

| Roll No | Subject | Marks |
|---|---|---|
| 1 | Math | 98 |
| 1 | English | 87 |
| 2 | Math | 81 |
| 2 | English | 92 |

In the above figure, the details of the students are stored in Student table. The Marks table stores only the Roll No of students. The remaining data is not duplicated. Roll No in the Marks table is duplicated for joining two tables.

## 2. Data Consistency

An important benefit of controlling redundancy is that the data is consistent. If a data item appears only at one place, it is easy to maintain. If it is required to update data, the updation is performed at only one place. The change will automatically take effect at all places where ever this data is used.

## 3. Consistency Constraints

Consistency constraints are the rules that must be followed to enter data in database. If the constrainsts are not fulfilled, data cannot be entered in database. Database management systems provide an easy way to apply different consistency constraints to ensure data consistency. For example, a constraint can be applied to ensure that the data is always entered in a specific range etc.

## 4. Data Atomicity

A collection of all steps to complete a process is known as **transaction**. Data atomicity means that either a transaction should take place as a whole or it should not take place at all. It ensures that the database will always have correct and consistent data. Suppose a user wants to transfer money from account A to account B. This process consists of two steps:

1. Deduct the money from account A.
2. Add the money to account B.

Suppose that the system fails when the computer has performed the first step. It means that the amount has been deducted from account A but has not been added to account B. This situation can make data inconsistent. The database management system does not allow such a situation to happen. Database management system either executes both steps or does not execute any step.

## 5. Data Security

Data security is the protection of the database from unauthorized access. The database management system provides several procedures to maintain data security. The security is maintained by allowing access to the database through the use of passwords. Not every use of database system should be able to access all the data.

In some situations, it is required to provide different types of access permission to data for different users. For example, a data entry operator should only be allowed to enter data. The chairman of the organization should be able to access or delete the data completely. Database management system provides different levels of security options for different users.

### 6. Reduced Development Time

A database organizes data more efficiently than a file processing system. It is often easier and faster to develop programs that use this data. Many database management systems also provide several tools to assist in program development. So it reduces the overall time for developing applications.

### 7. Compactness

The database management system stores data with compactness and efficiency. It requires less storage space than the file system. It saves the storage resources of the system and memory is not wasted.

### 8. Easier Reporting

Reports are very important part of database applications. The reports are very essential for taking crucial decisions in an organization. The data in database is stored in an organized manner. It can easily be retrieved for creating different reports. The reports can be prepared very easily and quickly in required format in database management system.

### 9. Data Sharing

Once a database is developed, it can be used by several users in the organization. The database can also be shared by different applications. If a new application requires the same data, it can share the existing database instead of developing it again.

### 10. Increased Concurrency

In some situation, two or more users may access the same file simultaneously. It is possible that the accesses will interfere with each other. This may result in loss of information or even loss of integrity. Many DBMS manage concurrent access and ensure such problems cannot occur.

### 11. Improved Backup and Recovery

In file-based systems, it is the responsibility of the user to protect data from failures of the computer system or application program. This may require taking backup of the data daily. If the data is lost, the backup is restored. The modern DBMS provide facilities to minimize the amount of processing that can be lost due to a failure.

### 12. Data Independence

Database approach provides the facility of data independence. It means that the data and the application programs are separate from each other. It is possible to change data storage structures and operations without changing the application programs.

## 1.9.2 Disadvantages of Database Approach

Some disadvantages of using database approach are as follows:

### 1. High Cost of DBMS

A complete database management system is very large and sophisticated software. It is expensive to purchase database management software.

### 2. Higher Hardware Cost

Database management systems are complicated and heavy softwares. Additional memory and processing power may be required to run the DBMS. It may require more powerful hardware.

### 3. Higher Programming Cost

DBMS is complex software with many features. The programmers need a thorough knowledge of system to use it to best advantage. If the organization hires experienced database programmers, it has to pay extra cost for this expertise.

### 4. High Conversion Cost

If an organization converts its records to database, data has to be converted from files to database system. Because of the different formats used by different systems, it may be a difficult and time-consuming process. Moreover, the structure and data may also have to be modified according to the requirements of DBMS.

### 5. More Chance of Failure

In database management system, all resources and components are centralized. If any of these components fails, the whole system stops.

### 6. Complexity & Performance

Database management system is general-purpose software. A complete DBMS has to perform many tasks that make it complex and complicated software. In some applications, DBMS may run less efficiently as compared to file processing system.

## 1.10 Difference between File and Database Approach

The difference between File and Database approach is as follows:

| File-based approach | Database approach |
|---|---|
| 1. The programs and data are inter-dependent. | 1. The programs and data are independent of each other. |
| 2. The data may be duplicated in different files that cause data redundancy. | 2. The data is not duplicated and appears only once. |
| 3. The same data in different file may be different that creates inconsistency. | 3. The data appear only once so it is always consistent. |
| 4. The data is separately stored in various files and it is difficult for applications to retrieve the appropriate data. | 4. The data is stored in tables which are linked together. The applications can retrieve the required data easily. |
| 5. The data is distributed in many different files and cannot be shared. | 5. The data is stored at one place and can be shared easily. |
| 6. It is difficult to apply data integrity checks on files. | 6. Database approach provides many constraints for data integrity. |
| 7. It provides poor security as the data is widely spread. | 7. It provides many procedures to maintain data security. |
| 8. It is difficult to maintain as it provides less controlling facilities. | 8. It provides many facilities to maintain the programs easily. |
| 9. It is less complex system. | 9. It is very complex system. |
| 10. The cost is very less than DBMS. | 10. The cost is much more than file system. |
| 11. One application can fail without affecting the others. | 11. All application relying on database fail if the database fails. |

## 1.11 Application Program

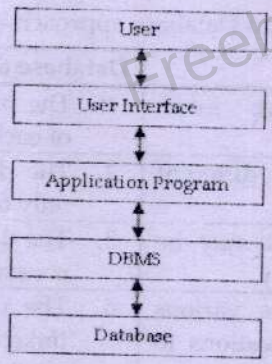
An application program is a program that is used to send commands to the database management system to manipulate database. These commands are sent to DBMS through graphical user interface. The user interacts with application program. The application program further interacts with database management system. Two important application programs are Developer2000 and Power Builder.

### 1.11.1 Relationship of Application Program & DBMS

A database application is developed by using both application program and database management system. The application program contains the user interface. The user uses this interface for communicating with the database management system to manipulate database. The application program is also called the **front-end** and the database is known as **back-end**.

The relation of application program and database management system is very important. When a database program is developed, a way of communication with the program is required for the user. The user cannot use the database directly. An easy and simple interface is required so that the user can easily use the database.

Another important relationship of DBMS and application program is that it is used to produce effective and informative report in application programs. These reports are very important for any organization for taking different decision about the business.




## 1.12 Range of Database Applications

The range of database applications is as follows:

### 1.12.1 Personal Computer Databases

Personal computer databases are specially designed for single user. The user may be using stand-alone desktop computer or laptop. The representatives of a medical company may use laptop computers when working in the market. This computer may use a simple database application in which the records of customers are stored. Some important decisions about the development of personal computer database applications are as follows:

- It should be purchased or developed within the organization?
- It should be developed by end user or a professional in information system?
- Required data and the design of database
- Selection of DBMS for database application
- Synchronization of personal computer database with other databases
- Accuracy of data in personal computer database

Personal computer databases are widely used. A problem with this database application is that it is difficult to share quickly. For example, if the head of medical company wants to get complete information of his customers, it cannot be done quickly as it is stored on the laptop computers of different representatives.

## 1.12.2 Workgroup Databases

A **workgroup** is a team of people who work on the same project collectively. A workgroup normally consists of less than 25 people. A type of database that is specially designed to support workgroups is known as **workgroup database**.

The people in workgroup are connected with one another through **local area network (LAN)**. The database is stored on a central computer called **server**. All members of the workgroup can share this database.

Workgroup databases provide many facilities but arise some problems also. The main problems are of security and integrity of data when the data is updated concurrently. Some important decisions about workgroups databases are as follows:

- Database optimization to satisfy requirements of different members of workgroup
- Concurrency control
- Decision about the location of different operations i.e. on server or workstation

## 1.12.3 Department Databases

A functional unit in an organization is known as **department**. An organization may have marketing, production and accounting departments. A department normally consists of 25 to 100 people. Department databases are specially designed to support the functions of a department. Some important decisions about department databases are as follows:

- Database design for efficient performance to handle a large number of users and transactions
- Proper security to protect data against unauthorized access
- Database tools for complex environment
- Ensure data redundancy and consistency if data is used in different departments
- The need for distributed database if users are geographically away from one another

## 1.12.4 Enterprise Databases

A type of database that is specially designed to support the functions of a whole organization or many departments of an organization is known as **enterprise database**. The most important enterprise database is known as **data warehouse**. The contents of a data warehouse are derived from many operational database like personal computer, workgroup and department databases.

Suppose a university has many departments like CS, Biology, Agriculture etc. Each department uses a department database to maintain the functions of department. The Vice Chancellor needs to get the information of different departments frequently. In order to satisfy this requirement, a data warehouse can be developed for the whole university in VC office. The data warehouse will extract the databases of different departments periodically.

Some important decisions about department databases are as follows:

- Distribution of data among various locations in the corporate structure
- Maintenance of standards for data names, definitions, and formats

## 1.13 Types of Users

There are different types of user that play different roles in a database environment. Following is a brief description of these users:

### 1. Application Programmers

Application programmer is a professional who writes computer programs in a high level language. These programs can be used to interact with databases. Application programmer designs application programs according to the requirements of the users. He works according to the specification provided by the system analyst.

### 2. End Users

End users are those persons who interact with the application directly. They are responsible to insert, delete and update data in the database. They get information from the system as and when required. Different types of end users are as follows:

- **Naive Users:** Naive users are the users who have no technical knowledge about the DBMS. They use database through application programs using simple user interface. They perform all operations by using simple commands provided in user interface. The data entry operator in an office is responsible for entering records in database. He performs this task by using menus and buttons etc. He does not know anything about database or DBMS. He interacts with database through application program.
- **Sophisticated Users:** Sophisticated users are the users who are familiar with the structure of database and facilities of DBMS. Such users can use a query language such as SQL to perform the required operations on databases. Some sophisticated users can also write application programs.

### 3. Database Administrator

Database administrator is the most technical user. He is responsible for managing the whole database system. He designs, creates and maintains the database. He manages the users who can access this database and controls integrity issues. Some important functions of a database administrator are as follows:

- Installation of software
- Monitoring of database system
- Solution of any problem that occurs in the database system
- Assigning permission to different users to use database system
- Taking regular backups of database
- Restoring the system in case of any problem or system crash

## 1.14 History of Database Systems

The concept of databases was introduced in 1960's. Since then, a lot of work has been done in this field. Following is a brief discussion about the history of database.

### 1960's

In 1960's, computers became cost effective for companies and the storage capability of computers increased. Two main data models were developed:

1. Network model (CODASYL)
2. Hierarchical (IMS)

Access to database was through low-level pointer operations. Storage details depended on the type of data to be stored. Thus adding an extra field to your database requires rewriting the underlying access/modification scheme. Emphasis was on records to be processed, not overall structure of the system. A user would need to know the physical structure of the database in order to query for information. One major commercial success was SABRE system from IBM and American Airlines.

### 1970-72

E.F. Codd proposed relational model for databases in a landmark paper on how to think about databases. He disconnects the schema (logical organization) of a database from the physical storage methods. This system has become standard since then.

### 1970's

Two main prototypes for relational systems were developed during 1974-77. These provide nice example of how theory leads to best practice.

- **Ingres:** Developed at UCB. This ultimately led to Ingres Corp., Sybase, MS SQL Server, Britton-Lee, Wang's PACE. This system used QUEL as query language.
- **System R:** Developed at IBM San Jose and led to IBM's SQL/DS & DB2, Oracle, HP's Allbase, Tandem's Non-Stop SQL. This system used SEQUEL as query language.

### 1976

P. Chen proposed **Entity-Relationship (ER)** model for database design. It gave another important insight into conceptual data models. Such higher-level modeling allows the designer to concentrate on the use of data instead of logical table structure.

### Early 1980's

In early 1980's, the commercialization of relational systems took place.

### Mid 1980's

SQL (Structured Query Language) became a standard. IBM launched DB2. The importance of network and hierarchical models was decreased. The development of the IBM PC gave rise to many DB companies and products such as RIM, RBASE 5000, PARADOX, OS/2 Database Manager, Dbase III, IV (later Foxbase and then Visual FoxPro), Watcom SQL.

### Early 1990's

In early 1990's, much work was done on client tools for application development such as PowerBuilder (Sybase), Oracle Developer, VB (Microsoft), etc. Client-server model for computing became the norm for future business decisions. The work on Object Database Management Systems (ODBMS) prototypes also started during this period.

### Mid 1990's

The usable Internet/WWW appeared in the middle of 1990. It allowed remote access to computer systems. The concept of Web/DB started to grow.

### Late 1990's

Internet companies worked for Web/Internet/DB connectors. Examples are Active Server Pages, FrontPage, Java Servlets, JDBC, Java Beans, ColdFusion, Dreamweaver, Oracle Developer etc. Open source solution came online with popular use of cgi, Apache, MySQL etc. **Online Transaction Processing (OLTP)** and **Online Analytic Processing (OLAP)** comes of age with many merchants using point-of-sale (POS) technology on a daily basis.

## Early 21st Century

The growth of DB applications continued in the early 21st century. More interactive applications appeared with use of PDAs. The main companies that predominated the large DB market are IBM, Microsoft, and Oracle.

## Future Trends

Huge (terabyte) systems are appearing and the handling and analyzing data has become very complex. Large science databases such as genome project, geological, national security, and space exploration data have been developed. Data mining, data warehousing, data marts are commonly used techniques today. This trend will continue in the future.

XML with Java has become a popular technique. Mobile database is now coming to market in various ways. Distributed transaction processing is also becoming very popular for business planning in many areas.

---

## Short Questions

**Q.1. Define data.**

A collection of raw facts and figures is called data. Data is collected from different sources and is not very meaningful for making decisions. Data may consist of numbers, characters, symbols or pictures etc. Data is collected for different purposes.

**Q.2. Define information.**

The processed data is called information. It is an organized and processed form of data.

**Q.3. Define database.**

Database can be defined as an organized collection of related data. The word "organized" means that data is stored in such a way that the user can store, manipulate and retrieve data easily. The word "related" means that a database is normally created to store the data about a particular topic.

**Q.4. Define database management system.**

A database management system (DBMS) is a collection of programs that are used to create and maintain a database.

**Q.5. Define security.**

The protection of the database from unauthorized users, which may involve passwords and access restrictions.

**Q.6. Define data integrity.**

The reliability and accuracy of data is called data integrity. The maintenance of the validity and consistency of the database by use of particular constraints that are applied to the data.

**Q.7. List four examples of database systems.**

- A system that maintains component part details for a car manufacturer
- An advertising company keeping details of all clients and adverts placed with them
- A training company keeping course information and participants' details
- An organization maintaining all sales order information

**Q.8. State the purpose of a database.**

The purpose of a database is to help people and organizations keep track of things.

**Q.9. What is metadata? Give an example.**

Data about the structure of a database is called metadata. Examples of metadata are the names of tables, names of columns and tables, properties of tables and columns etc.

**Q.10. Briefly describes the history of database processing.**

In file processing, data was maintained in sequential lists on magnetic tape. Database processing became possible with the availability of direct access disk storage in 1960s. Using this storage, both the hierarchical data model and network data model were developed. In 1970, E. E. Codd of IBM proposed relational model that is the standard model used today. Current DBMSs such as DB2, Oracle and SQL Server are based on relational model. More recent events include the appearance of microcomputer based DBMSs, introduction of Object Oriented DBMS and development of tools such as XML used with database systems over the Internet.

**Q.11. What does DBMS stand for? List the functions of a DBMS.**

DBM DBMS stands for Database Management System. The functions of the DBMS are:

- Create Database
- Create Tables
- Create Supporting Structures
- Read Database Data
- Update Database Data
- Maintain Database Structures
- Enforce Rules
- Control Concurrency
- Provide Security
- Perform Backup and Recovery

**Q.12. Contrasts the following terms:**

Structural Dependence and Data Dependence
Data dependence and data independence
Repository and database

**Structural dependence and data dependence:** Any chyange in the structure of a file such as addition or deletion of a field requires the modification of all programs using that file. This modification is necessary becase the access to a file is dependent on its structure. This type of dependence is known as *structural dependence*. Any change in file data characteristics such as changing a field from integer to decimal requires modification of all programs that access the file. This type of dependence is known as *data dependence*.

**Data dependence and data independence:** Data dependence is a tight relationship between data and specific programs. The application programs has to be changed if there is any change data. Data independence means that data and application programs are separate from each other. The data storage structures and operations can be changed without changing application programs.

**Repository and Database:** A repository is a centralized storehouse of all data definitions, relationships and other system components. A database is an organized collection of logically related data.

**Q.13. Why have databases become preferred method to store data used by an information system?**

Databases are a common point of access, management and control. They allow data to be managed as an enterprise-wide resource. They simultaneous access to many users and application programs. They solve many problems associated with separately maintained data stores such as redundancy, inconsistent security and inconsistent data access methods.

**Q.14. Give an example of a large-enterprise database application.**

A construction company uses a database to keep track of project costs, labor, materials and schedule. One database supports all of these different applications.

**Q.15. What were some of the weaknesses of early organizational database applications?**

The earily organizational database applications were slow and unreliable. DBMS developers did not know efficient ways to provide database access. The programmers did not know how to use new database technology.

**Q.16. What is the major drawback of personal databases?**

The major drawback of personal databases is that their data cannot be shared with other users. For this reason, they should be limited to special situations where there is not a need to share the data.

**Q.17. How is relationship between application programs and DBMS changing over time?**

DBMS is gradually taking on more and more of the application programs functions and roles.

**Q.18. What was the major factor that gave rise to workgroup database applications?**

The major factor that gave rise to workgroup database is the development and acceptance of LAN technology and products.

**Q.19. List different categories of databases with example of each type.**

a. **Personal computer databases:** It is a set of data that describes patient visits, recorded by a home health-care professional.
b. **Workgroup database:** It is a database that supports the work of several scientists performing research on a new drug.
c. **Department database:** It is a database used by human resources department of large hospitals.
d. **Enterprise database:** It is a database that supports SAP enterprise information system.

**Q.20. Which problems does data redundancy cause?**

Redundant data uses additional storage space and makes it difficult to maintain the accuracy of database when changes are made.

**Q.21. Why do people use databases for information-handling tasks?**

(1) Databases make it easier to store large quantities of information.
(2) Databases make it easier to retrieve information quickly and flexibly.
(3) Databases make it easy to organize and reorganize information.
(4) Databases make it easy to print and distribute information in many ways.

**Q.22. What is the function of application metadata? How does it differ from metadata?**

Application metadata is a description of application components such as forms, reports, menus and queries. It differs from metadata that is a description of the database structure.

**Q.23. Explain the purpose of forms, reports, queries and menus.**

A form is used to enable users to create, read, modify and delete data. A report is a structured presentation of database data. Queries allow the users to answer questions from the data. Menus are structured presentations of allowed user actions.

**Q.24. List and describe some of the problems of the traditional file environment.**

Following are some problems with traditional file environment:

1. **Data redundancy** is the presence of duplicate data in multiple data files. In this situation, confusion results because the data can have different meanings in different files.

2. **Program-data dependence** is the tight relationship between data stored in files and specific programs required. This dependency is very inefficient, resulting in the need to make changes in many programs when a common piece of data changes.

3. **Lack of flexibility** means that it is very difficult to create new reports from data when needed.

4. **Poor security** results from the lack of control over the data because the data are so widespread.

5. **Data sharing** is virtually impossible because it is distributed in so many different files around the organization.

**Q.25. List and describe each of the components in the data hierarchy.**

The data hierarchy includes bits, bytes, fields, records, files and databases. Data are organized in a hierarchy that starts with bit. A bit is represented by either a 0 (off) or a 1 (on). Bits can be grouped to form a byte to represent one character, number or symbol. Bytes can be grouped to form a field such as a name or date. Related fields can be grouped to form a record. Related records can be collected to form files. Related files can be organized into a database.

**Q.26. Name the key personnel involved with databases and briefly describe their roles**

The database administrator has both technical and managerial responsibilities over the database resource. Database programmers are required to create efficient data processing computer code. The end users have a major impact on database design, use and efficiency.

---

## Multiple Choice

1. **A collection of raw facts and figure is called:**
   a. Data b. Information c. Processing d. None
2. **The manipulated and processed data is called:**
   a. Object b. Information c. Data d. None
3. **Manipulation of data to achieve the required objectives and result is called:**
   a. Data processing b. Operation c. Both a and b d. None
4. **A person's account, car, and house are considered:**
   a. Object b. Table c. Data processing d. None
5. **A collection of related fields is:**
   a. File b. Record c. Database d. None
6. **All records in a file have the same:**
   a. Contents b. Structure c. Both a and b d. None
7. **A collection of data that consists of name, address and email of a person is called:**
   a. Byte b. Record c. Character d. Field
8. **A record in a database is the information referring to a:**
   a. person b. Product c. Event d. All
9. **Each item of information within a record is called:**
   a. File b. Field c. Both a and b d. Byte
10. **A logical grouping of characters is a:**
    a. Field b. Record c. File d. All
11. **A field is to a record as:**
    a. Data are to files b. A column is to a row
    c. Files are to tables d. Attributes are to columns
12. **A database containing all students in a class would store basic data of students in:**
    a. Record c. Field d. Cell d. File
13. **A database containing all students in a class would store RollNo of a student in:**
    a. Record b. Field d. Cell d. File
14. **A database containing all students in a class would store the information of individual students in:**
    a. Record c. Field d. Cell d. File

**15. Which of the following is also known as data set?**
a. Record b. Field c. File d. All

**16. A set of related files created and managed by a (DBMS) is called:**
a. Field b. Record c. Database d. None

**17. Which of the following is an example of a database?**
a. Phone book b. Library catalog c. Student records d. All

**18. SQL is a(n):**
a. Unstructured language b. Structured Language
c. Object oriented language d. Software

**19. SQL stands for:**
a. Sort-Query-List b. Self-Quantifying-Language
c. Seek-Qualify-Label d. None

**20. SQL can be used to:**
a. Create database structures only b. Query database data only
c. Modify database data only d. All

**21. A database is an organized collection of related data.**
a. Logically b. Physically c. Loosely d. Badly

**22. The objectives of database management systems include:**
a. Database Integrity b. Data Integration c. Availability d. All

**23. Information sharing means that:**
a. Information can be stored once and retrieved any number of times.
b. The same information can be shared by different applications.
c. Both a and b
d. Neither a nor b

**24. Which of the following is handled by DBMS?**
a. Data integrity b. Data security c. Data independence d. All

**25. A program whose job is to store and retrieve user data in the database is called:**
a. Database Modeling System b. Database Management System
c. Data Business Model System d. Data Business Management Service

**26. The database system is composed of four major parts:**
a. Hardware, Hard drive, Monitor, Data, User
b. Hardware, Software, People and Data.
c. Software, You, Me, DBA, Client
d. DBMS, Hardware, User, Programmer, Engineer

**27. In a database processing system:**
a. The database application(s) interact with the DBMS
b. The database application(s) access the database data
c. The DBMS accesses the database data
d. a and c

**28. Data that causes inconsistency lacks:**
a. Good data b. Data integrity c. Data redundancy d. Data anomaly

**29. Which one of the following is an advantage of database management approach?**
a. Programs are independent of the data format.
b. Reduced security and control of the data.
c. Increased duplication of data. d. All

**30. DBMS stands for:**
a. Database Modeling System b. Database Management System
c. Data Business Model System d. Data Business Management Service

**31. Which of the following is NOT an advantage of database systems?**
a. Redundant data b. Data independence
c. Backup /Recovery d. Better data quality

**32. Which of the following enables the user to modify data structures without affecting existing programs that use them?**
a. Data dependence b. Data independence
c. Data integration d. Data relationships

**33. Database application contain procedures for:**
a. Adding records b. Deleting records c. Processing queries d. All

**34. Which of the following is related to a component of DBMS known as personnel?**
a. Application programmer b. End users
c. Database administrator d. All

**35. The major component of DBMS is called:**
a. Database Manager b. File Manager c. Data Manager d. All

**36. Duplicate data in multiple data files is:**
a. Data redundancy b. Data multiplication c. Data Integrity d. None

**37. The description of structure and organization of data in a database is contained in:**
a. Data dictionary b. Data mine c. Structured query language d. None

**38. A request for information from a database in database terminology is called:**
a. Report b. Letter c. Table d. Query

**39. A printed or onscreen display of data or information in the database is called a(n):**
a. Entity b. Report c. Query d. Screen

**40. The name for software to build reports that summarize data from a database is:**
a. Report writer b. Reporter c. Report Builder d. Report Generator

**41. Which of the following is not a database management system?**
a. MS Access b. MS SQL c. Oracle d. None

**42. For database systems needing to support approximately 15 concurrent users within an organization, which type of database would be appropriate?**
a. Internet databases b. Organizational databases
c. Workgroup database d. Personal database

**43. For database systems needing to support approximately 2 trillion bytes of data within an organization, which type of database would be appropriate?**
a. Internet databases b. Organizational databases
c. Workgroup database d. Personal database

## Answers

| | | | | | |
|---|---|---|---|---|---|
| 1. a | 2. b | 3. c | 4. a | 5. b | 6. b |
| 7. b | 8. d | 9. b | 10. a | 11. b | 12. d |
| 13. b | 14. a | 15. c | 16. c | 17. d | 18. b |
| 19. d | 20. d | 21. a | 22. d | 23. c | 24. d |
| 25. b | 26. b | 27. d | 28. b | 29. a | 30. b |

| | | | | | |
|---|---|---|---|---|---|
| 31. a | 32. b | 33. d | 34. d | 35. a | 36. a |
| 37. a | 38. d | 39. b | 40. d | 41. b | 42. c |
| 43. b | | | | | |

## True / False

1. DATA and INFORMATION are indistinguishable terms.
2. The management of metadata is more important than management of the actual data.
3. Organizations that rely on traditional file processing systems may devote as much as 80 percent of the total information systems budget to program maintenance.
4. PDA or laptop computer will likely contain a workgroup database.
5. Personnel, marketing, manufacturing, and accounting are all examples of workgroups and use workgroup databases.
6. A multi-user database often requires substantial personnel training on an ongoing basis.
7. A repository is a centralized knowledge base for all historical data.
8. The first database management systems were introduced during the 1970s.
9. E.F. Codd and others developed the hierarchical DBMS model in the 1960s.
10. A database program can appropriately be characterized as a computerized file cabinet.
11. An advantage of using a database instead of paper is that it makes it easier to organize information.
12. A database file is a collection of fields.
13. Records are combined to form a file.
14. Files are combined to form records.
15. A field containing the number of hours worked would be a numeric field.
16. A request of information from a database is called search string in database terminology.
17. The standard language supported by most databases for programming complex queries is called SQL.
18. The purpose of a database is to help people keep track of things.
19. The typical size of a workgroup database is less than 100 megabytes.
20. An organizational database is a database that typically has less than 100 concurrent users.
21. A file processing system generally has less data duplication than a database system.
22. A problem with file processing systems is that it is difficult to represent data in the users' perspective.
23. A major problem with a database system is that application programs are generally dependent on file format.
24. A collection of data has data integrity if the data are logically consistent.
25. With file processing systems, application programs do not generally depend on the file formats.
26. In a file processing system all the application data are stored in a single facility called the database.

27. One of the consequences of program data dependency is that file formats depend on the language or product used to generate them.
28. Distributed databases eliminate most of the problems of security and control that are seen in other database systems.
29. A DBMS is software.
30. Database record may be entered and modified but not deleted.
31. In a database, field name must be unique.
32. There is exactly one file per database.
33. The fields in a particular database record contain unrelated data.
34. Users can use a database successfully only if they understand the underlying technology.
35. One disadvantage of databases is that they can use by only a single user.
36. In a traditional file environment, any change in data requires a change in all programs that access the data .
37. DBMS separate the logical and physical views of the data.
38. Distributed systems reduce the vulnerability of a single central site.
39. A traditional system serves a wider community of users than a database because less training is involved.
40. Database management systems enable organizations to analyze information easily.
41. The main purpose of a database management system is to develop interactive Web pages.
42. A popular personal computer DBMS is Microsoft Access.
43. In the data hierarchy, a file is a collection of data fields.
44. Databases store data in single tables to reduce data redundancy.
45. The role of the end user is to maintain a secure database.
46. In a database processing system, rules are processed by the database management system.
47. A database is called self-describing because it reduces data duplication.
48. An index can be used to improve the performance of the database.

## Answers

| | | | |
|---|---|---|---|
| 1. F | 2. T | 3. T | 4. F |
| 5. F | 6. T | 7. F | 8. F |
| 9. F | 10. T | 11. T | 12. F |
| 13. T | 14. F | 15. T | 16. F |
| 17. T | 18. T | 19. T | 20. F |
| 21. F | 22. T | 23. F | 24. T |
| 25. F | 26. F | 27. T | 28. F |
| 29. T | 30. F | 31. T | 32. F |
| 33. F | 34. F | 35. F | 36. T |
| 37. T | 38. T | 39. F | 40. T |
| 41. F | 42. T | 43. T | 44. F |
| 45. F | 46. T | 47. F | 48. T |

# Database Environment

## Chapter Overview

## 2.1 Three-Level Architecture

**Database Task Group (DBTG)** developed and published a proposal for a standard vocabulary and architecture for database systems in 1971. It was appointed by **Conference on Data Systems and Languages (CODASYL)**. The **Standards Planning and Requirements Committee** of **American National Standards Institute (ANSI) Committee on Computers and Information Processing** developed and published a similar vocabulary and architecture in 1975.

The result of these reports was the **three-level architecture**. Three-level architecture is the basis of modern database architecture. Database can be viewed at three levels. The three levels are depicted by three models known as **three-level schema**. The models refer to the structure of the database not the data stored in it. The permanent structure of database is known as **intension of database** or **database schema**. The data stored at a given time is known as **extension of database** or **database instance**.

The intension of a database should not be changed once it has been defined. This is because a small change in the intension of database may require many changes to the data stored in database. The extension of database is performed after the intension of database has been finalized. It means that data is stored in database when database structure has been defined. The extension of the database is performed according to the rules defined in the intension of database.

The schemas are used to store definitions of the structures of database. It can be any thing like a single entity or the whole organization. Three-level architecture defines different schemas stored at different levels to isolate the details of different levels from one another.

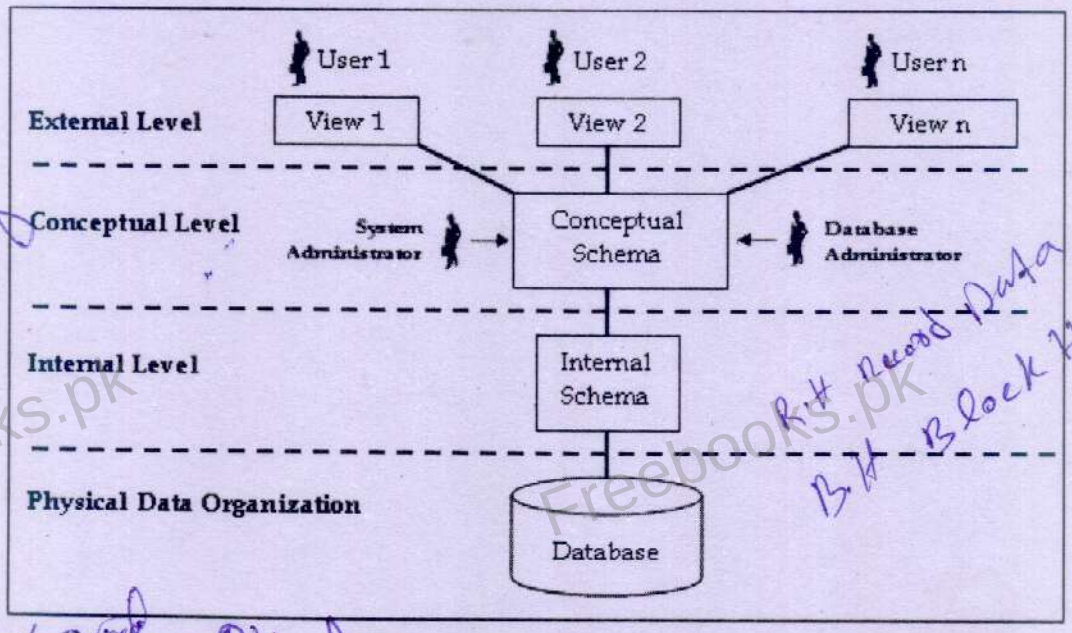



**Figure:** Three-Level Architecture

i) internal/conceptual
ii) Conceptual/external

### 2.1.1 External Level / View

The external level consists of different external **views** of a database. Each external view is the view of a particular user about the system. Different users conceive the system in different ways. Each user is interested in one part of the system and ignores other parts. One user may not be aware of the whole system at all.

Different views may have different representations of the same data. For example, one user may think that dates are stored in the form (month/day/year). Another user may think they are represented as (year/month/day).

Some views might include **virtual** or **calculated data**. Virtual data is the data that is not actually stored in database but created when needed. For example, the product name and its price can be stored in database. The total bill can be created as and when desired. Similarly, the marks of different subjects can be stored in database. The overall grade of the students can be calculated when result card is prepared.

External level is described in **external schema**. It is also called **subschemas**. External schemas refer to different views of data.

### 2.1.2 Logical or Conceptual Level / View

This is the middle-level view in three-level architecture. The logical or conceptual level describes the data stored in the database. It contains the definition of the data to be stored in the database. It also contains the rules and information about the structure and type of data. It is the complete description of data stored in database. That is why it is also known as **community view** of the database.

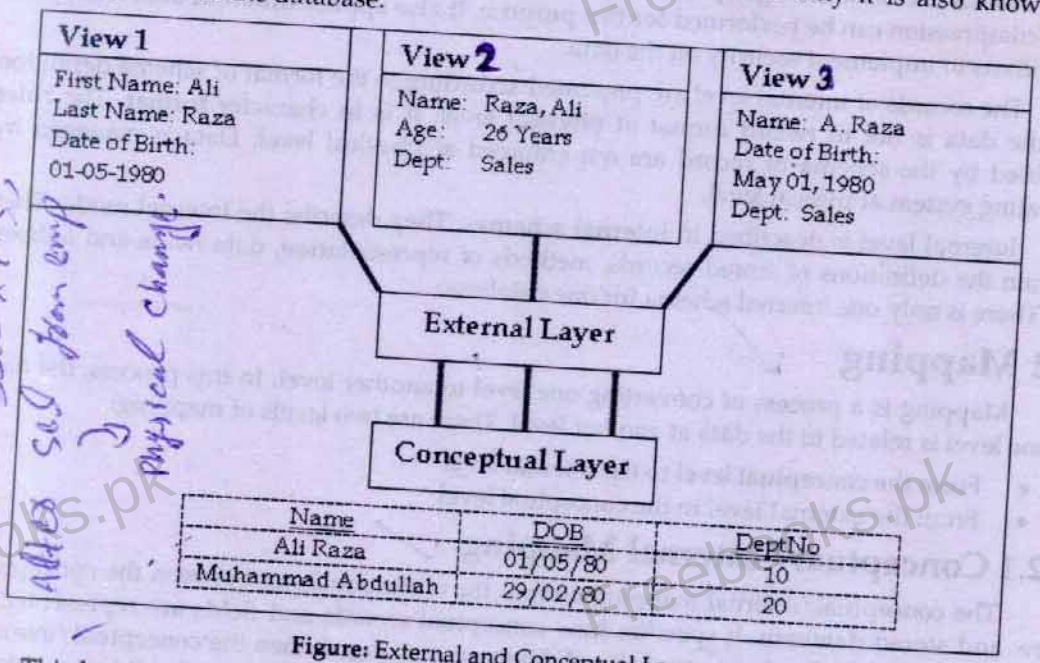



**Figure:** External and Conceptual Layers

This level contains the logical structure of entire database as seen by DBA. It hides the details of physical storage structure. The conceptual level represents the following:

- All entities, their attributes and their relationship
- The constraint on the data

- Semantic information about the data
- Security and integrity information

The conceptual level supports each external view. It means that any data required by any user must be available from conceptual level. The conceptual model is comparatively constant. DBA designs a conceptual model to fullfil the present and future requirements of the organization. If there is any change in external model, the conceptual model should be able to accomodate that change. It is important because any change in the conceptual model requires a lot of effort. It also affects other views or levels of the database. The conceptual level is described in **conceptual schemas**. There is only one schema for one database.

### 2.1.3 Internal or Physical Level

Internal level is responsible to store data on storage media. It describes the physical representation of database on computer. It describes how the data is stored in database. It covers the data structures and file organization used to store data on storage devices.

Internal and physical levels are normally considered to be same. But there is a slight difference between them. The data is stored in binary format on disks. The binary storage method is implemented by operating system. DBMS partially decides the way data is stored on the disk. This decision of DBMS is based on the specifications of DBA. Additionally, DBMS adds information to the data to be stored. For example, it selects a specific file organization for storing data on disk. It also creates specific indexes to implement that file system. It uses the same index information for retrieving the data from the disk.

DBMS performs storage space utilization to consume minimum space for storing data. Data compression can be performed for this purpose. It also applies different data encryption algorithms to implement security on the data.

The records at internal level are presented according to the format of schema definition but the data is not in record format at physical level. It is in character format. The rules specified by the schema of record are not enforced at physical level. Data is managed by operating system at pysical level.

Internal level is described in **internal schemas**. They describe the internal mode. They contain the definitions of stored records, methods of representation, data fields and indexes etc. There is only one internal schema for one database.

## 2.2 Mapping

Mapping is a process of converting one level to another level. In this process, the data at one level is related to the data at another level. There are two levels of mapping:

- From the conceptual level to the internal level
- From the external level to the conceptual level

### 2.2.1 Conceptual / Internal Mapping

The conceptual/internal mapping defines the correspondence between the conceptual view and stored database. It specifies how conceptual records and fields are represented at the internal level. If the structure of stored database is changed, then the conceptual/internal mapping must be changed accordingly so that the conceptual schema can remain consistent. It is the responsibility of DBA to manage such change.

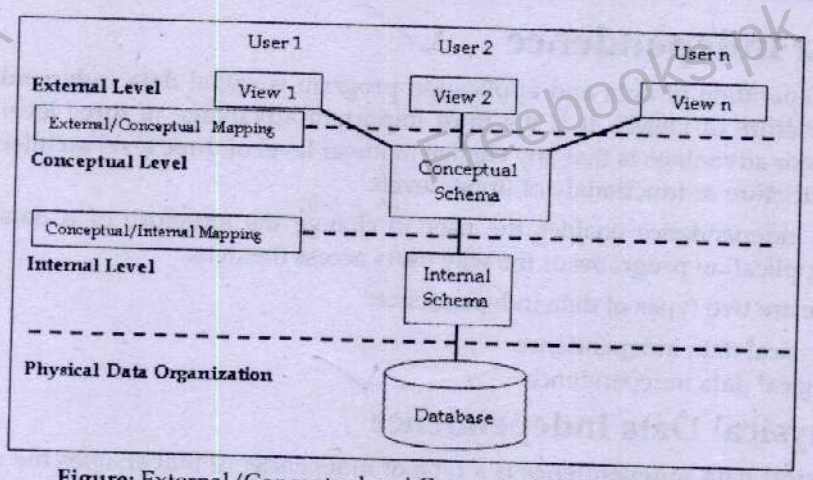



**Figure:** External/Conceptual and Conceptual/Internal Mapping

## 2.2.2 External/Conceptual Mapping

An external/conceptual mapping defines the correspondence between a particular external view and conceptual view. Generally, the differences between these two levels are similar to the differences between conceptual view and stored database. For example, fields can have different data types, fields and record names can be changed, several conceptual fields can be combined into a single field. Any number of external views can exist at the same time. Any number of users can share a given external view and different external views can overlap each other.

The following figure shows the represenation of data at different levels of database architecture. The data is stored in binary format at physical level. It is separate from internal view of data. The data is prefixed with **Block Header (BH)** and **Record Header (RH)**. The record header is used with every record. The block header is used with a group of records.

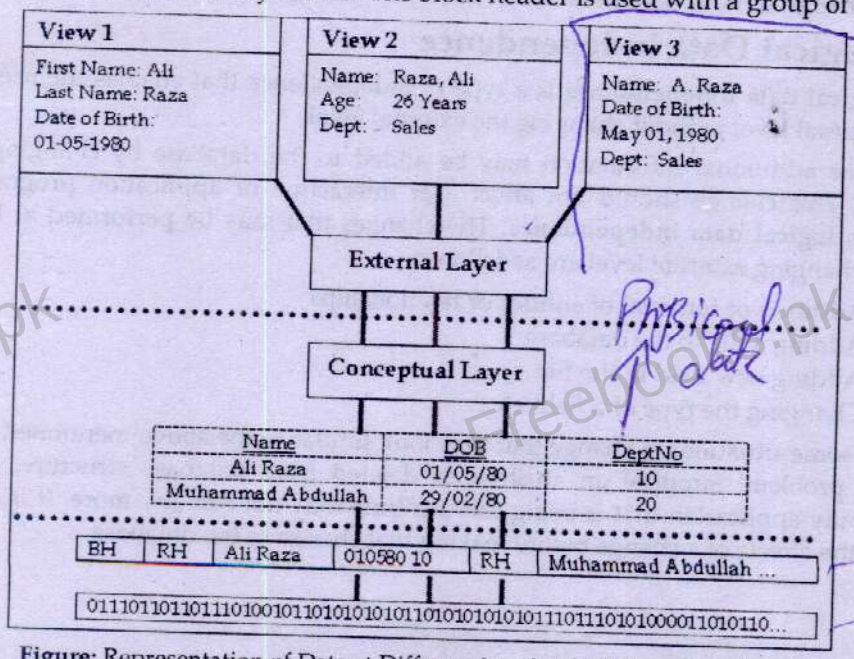



**Figure:** Representation of Data at Different Levels of Database Architecture

## 2.3 Data Independence

The separation of data and application program is called **data independence**. It is an important feature of DBMS. It is the most important advantage of three level architecture. Another major advantage is that any change in lower level of three level architecture does not affect the structure or functionaly of upper levels.

Data independence enables the user to change the structure of a database without changing application programs or the way users access the data.

There are two types of data independence:

- Physical data independence
- Logical data independence

### 2.3.1 Physical Data Independence

**Physical data independence** is a type of independence that enables the user to change the internal level without changing the conceptual level. In a DBMS, physical structure of database may change without changing application programs or altering the user's view of data. It is possible because DBMS uses **abstraction**. Data is translated from the way it is physically stored on disk to the representation and access techniques used by logical view.

If the physical structure changes, DBMS is aware of these changes but still provides the same logical view. The logical view remains constant and the application programs and user interactions based on logical view of data are not altered. The changes that may be performed at physical level without changing logical level are as follows:

- Changing file organizations or storage structures
- Using different storage devices
- Modifying indexes
- Modifying hashing algorithms
- Changing the access method

### 2.3.2 Logical Data Independence

**Logical data independence** is a type of independence that enables the user to change the conceptual level without changing the external level.

Some additional information may be added to the database by changing its logical structure. This change should not affect user interaction or application programs. This is known as **logical data independence**. The changes that may be performed at logical level without changing external level are as follows:

- Addition or removal of entities or relationships
- Adding a file to the database
- Adding new field in the file
- Changing the type of a field etc.

In some situations, a change that may look similar to the above-mentioned changes can create a problem. Suppose an attribute is deleted from database structure. It is serious because any application that is using this attribute may not run any more. It is important to analyze the effects of a change before making that change to the database.

## 2.4 Data Models

A representation of real world objects, events and their associations is called a **model**. The model helps the user to understand the complexities of the reald world environemnt. A collection of concepts to describe and manipulate data, relationships between data and constraints on data is called **data model**.

### 2.4.1 Parts of Data Model

A data model consists of the following parts:

- **Structural Part:** It consists of a set of rules. These rules specify how a database can be developed.
- **Manipulative Part:** It defines the types of operations that can be performed on data. The operations include updating or retrieving data from database and changing the structure of database.
- **Set of Integrity Rules:** It ensures the accuracy of data in the database.

### 2.4.2 Importance of Data Model

The data model is used as a communication tool for database designer, application programmer and end user to interact with one another. A good data model enables the users to understand the organization for which the database design is developed. A good data model is very necessary to design a good database.

Any DBMS is based on a specific data model. No DBMS can exist without any data model. It is difficult to create a proper database without knowing the data model of DBMS. It is very important to know the structures, manipulation languages and integrity facilities implemented by DBMS. It enables the user to understand the facilities and functionalities provided by the DBMS.

## 2.5 Types of Data Models

Different types of data models are as follows:

1. Object-Based Data Models => consist on object characteristics
2. Record-Based Models
3. Physical Data Models

Object-based data models and record-based data models are used to describe data at conceptual and external levels. The physical data models are used to describe data at internal level.

### 2.5.1 Object-Based Data Models

Object-based data models use the concepts like entities, attributes and relationships. An **entity** is a person, place, thing or event for which data is collected and maintained in the database. An **attribute** is the characteristics of an entity. A **relationship** is an association between two or more entities. Some types of object-based data models are as follows:

- Entity-Relationship
- Semantic
- Object-Oriented etc.

In diagram is entity and Table is Physical.

## 2.5.2 Record-Based Data Models

Record-based models are basically used to describe external and conceptual levels of database. They can also be used to describe internal level to some extent. They are used to develop and specify the logical structure and provide some options for implementation of the design. In record-based data models, database consists of different records. The records may be of different types. Each record types defines a fixed number of fields.

There are three types of record-based data models:

- The Hierarchical Model
- The Network Model
- The Relational Model

### 2.5.2.1 Hierarchical Model

One of the earliest database management systems was based on hierarchical model. In this model, records have a **parent-child relationship**. One of the most popular hierarchical database management systems was **Information Management System (IMS).** It was introduced by IBM in 1968. IMS is still the most widely used DBMS on IBM mainframes.

Consider the application used for Production Planning in automobile manufacturing companies. The model of the database is shown in the following figure. The automobile manufacturer produces various models of cars. Each car model is decomposed in assemblies like Engine, Body and Chassis. Each assembly is further decomposed into sub-assemblies like valves, spark plugs and so on. If the manufacturer wants to generate the Bill of Materials for a particular model of an automobile, the hierarchical data model is suitable because the bill of materials for a product has a hierarchical structure. Each record represents a particular part. Since the records have a parent-child relationship, each part is linked to its sub-part.

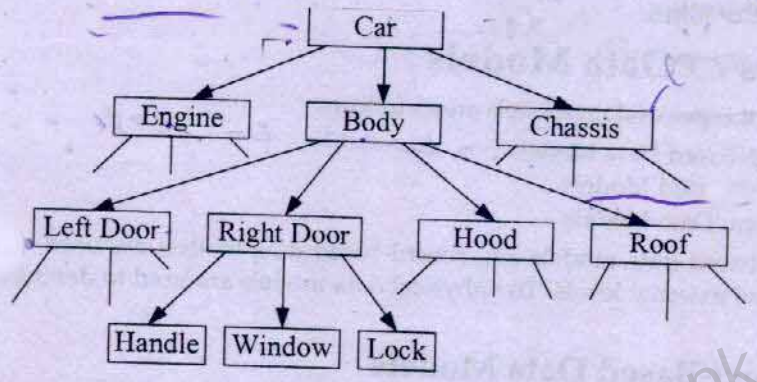



**Figure:** The Hierarchical Model

### Characteristics of Hierarchical Model

The characteristics of a hierarchical DBMS are as follows:

#### 1. Representation of Data as Hierarchical Trees

The hierarchical database is characterized by parent-child relationships between records. A record type R1 is called the parent of record type R2 if R1 is one level higher than R2 in the hierarchical tree. The root of the hierarchy is the most important record type. All records at different levels of the hierarchy are dependent on the root.

### 2. Each Sub-Module has only one Super-Module

Each child record (sub-module) has only one parent record (super-module). The parent record can have one or more children record types.

### 3. Represents a set of Related Records

There can be one or more record occurrences for a given record type. When the user writes into database, only one occurrence of record type is written. Similarly, when a record is retrieved from database, only one occurrence of record type is retrieved.

### 4. Hierarchy through Pointers

In a hierarchical database, pointers are used to link the records. Pointers determine whether a particular record occurrence is a parent or a child record. The path from the parent to the child is maintained through pointers.

### 5. Simple Structure

The database is a simple hierarchical tree. The parent and child records are stored close to each other on the disk. It minimizes disk input and output.

### 6. High Performance

The parent-child relationship is stored as pointers from one record to another. The navigation through the database is very fast resulting in high performance.

### 7. Predefined Relationships between Record Types

Record types at different levels of the hierarchy are dependent on the root. Root is the most important record type in the hierarchy.

### 8. Difficult to Reorganize

It is difficult to reorganize database because the hierarchy has to be maintained. Each time a record type is inserted or deleted, the pointers have to be manipulated to maintain the parent-child relationship.

### 9. More Complex Real Life Requirements

The hierarchical DBMS is based on a simple parent-child relationship. The real life applications are more complex and cannot be represented by hierarchical structure. For example, in an order-processing database, a single order might participate in three different parent-child relationships. It may link the order to customer who placed the order, the items ordered and sales person who took the order. This complex structure cannot be represented in a hierarchical structure.

## 2.5.2.2 Network Model

Network model was developed to overcome the problems of hierarchical data model. It modified the hierarchical model by allowing multiple parent-child relationships. These relationships are known as **sets** in network model. Its structures and language constructs were defined by **CODASYL (Conference on Data Systems Language)**.

### Characteristics of Network Model

The characteristics of a network DBMS are as follows:

### 1. Data Record Types are represented as Network

In this model, data record types are represented as a network.

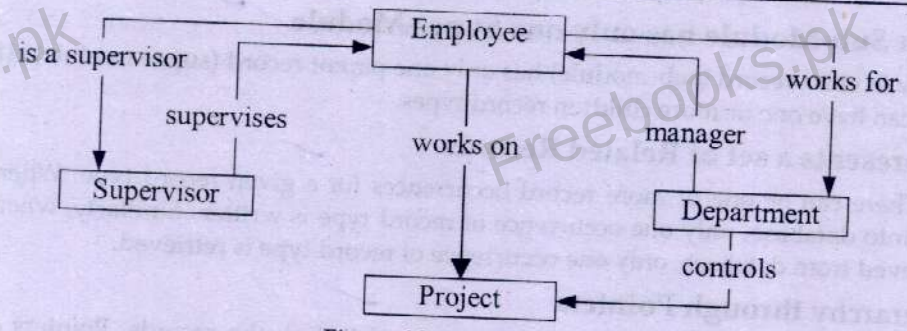



**Figure:** Network Model

### 2. Each sub-module can have one or more super-modules

Since multiple parent-child relationships are supported, child record type could have more than one parent record types.

### 3. Represents a set of related records

The sets that support multiple parent-child relationships and the structure of records have to be specified in advance.

### 4. Complex Structure

It supports multiple parent-child relationships that make database structure complex.

### 5. Relationships are Predefined

Network database implements sets that support multiple parent-child relationships. The sets have to be specified in advance. In the tradeoff between flexibility and performance, a network model is not very flexible to reorganize but has high performance levels.

### 6. Difficult to Reorganize

The network database is very difficult to reorganize. The insertion or deletion of record involves tracing the pointers and changing the appropriate links.

### 7. Navigation done by Programmer

The programmer has to write 3GL programs to specify the relationship and direction to navigate database. A record-by-record navigation is required to access a particular record.

### 8. 3GL Inadequate for Handling Sets

The records in a network model are processed one set at a time. 3GLs handle only one record at a time and hence are inadequate for handling the sets.

### 9. Query Facility not Available

Network database management systems do not have any query facility. 3GL programs have to be written specifying the path and the relationship.

## 2.5.2.3 Relational Model

Dr. E. F. Codd worked to improve the working of DBMSs to handle large volumes of data. He applied the rules of mathematics to solve the problems of earlier database models. Some important problems were as follows:

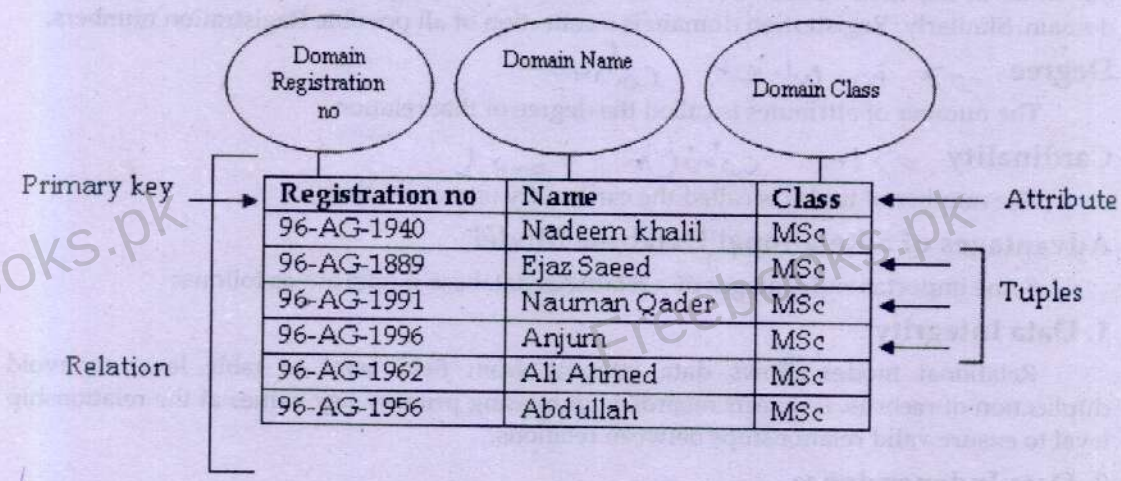
- Data Integrity
- Data Redundancy

Dr. Codd presented a paper **A Relational Model of Data for Large Shared Databanks** in June 1970 that contained 12 rules. A DBMS that satisfies these rules is called a **full Relational Database Management System (RDBMS)**. The term **relation** is also derived from the **set theory** of mathematics.

In a relational model, data is stored in relations. Relation is another term used for table. A table in a database has a unique name that identifies its contents. Each table can be called an intersection of rows and columns. An important property of a table is that the rows are unordered. A row cannot be identified by its position in the table. Every table must have a column that uniquely identifies each row in the table.

## Relational Database Terminology

Some important terminologies used in relational database model are as follows:




| Registration no | Name | Class |
|---|---|---|
| 96-AG-1940 | Nadeem khalil | MSc |
| 96-AG-1889 | Ejaz Saeed | MSc |
| 96-AG-1991 | Nauman Qader | MSc |
| 96-AG-1996 | Anjum | MSc |
| 96-AG-1962 | Ali Ahmed | MSc |
| 96-AG-1956 | Abdullah | MSc |

**Figure:** Database Terminology

## Relation

In a relational model, data is stored in relations. Relation is another term used for table. Following is an example of a relation.

| RegistrationNo | Name | Class |
|---|---|---|
| 96-AG-1940 | Nadeem Khalil | MSc |
| 96-AG-1889 | Ejaz Saeed | MSc |
| 96-AG-1991 | Nauman Qader | MSc |

**Figure:** An example of Relation

## Tuple

In a relational model, every relation or table consists of many tuples. Tuples are also called records or rows.

| | | |
|---|---|---|
| 96-AG-1940 | Nadeem Khalil | MSc |

| | | |
|---|---|---|
| 96-AG-1991 | Nauman Qader | MSc |

**Figure:** Two tuples of a Relation

### Attributes

An attribute is a named column of a relation. Attributes are also called characteristics. The characteristics of the tuple are represented by attributes or fields.

| |
|---|
| 96-AG-1940 |
| Ejaz Saeed |
| MSc |

**Figure:** Three attributes of a Relation

### Domain

A domain is a collection of all possible values of one or more attributes. For example, the value in the field "Class" can be the name of any taught classes. It is known as class domain. Similarly, Registration domain is a collection of all possible Registration numbers.

### Degree => in which colum

The number of attributes is called the degree of that relation.

### Cardinality => In which rows.

The number of tuples is called the cardinality of that relation.

### Advantages of a Relational Database Model

Some important advantages of a relational database model are as follows:

#### 1. Data Integrity

Relational model allows data integrity from field level to table level to avoid duplication of records. It detects records with missing primary key values at the relationship level to ensure valid relationships between relations.

#### 2. Data Independence

The implementation of database will not be affected by changes made in the logical design of the database or changes made in the database software.

#### 3. Structural Independence

**Structural independence** exists when the structure of database can be changed without affecting DBMS's ability to access the data. The relational database model does not use a navigational data access system. The data access paths are irrelevant to relational database designers, programmers and end users. Any change in relational database structure does not affect data access in any way. It makes relational databases model **structure independence**.

#### 4. Data Consistency & Accuracy

Since multiple level check and constraints are built-in, data is accurate and consistent.

#### 5. Easy Data Retrieval & Sharing

Data can be easily extracted from one or multiple relations. Data can also be easily shared among users.

## 2.5.3 Physical Data Models

Physical data models describe storage of data in computer. They represent information such as record structures, record orderings and access paths. There are not as many physical data models as logical data models. The most common physical data models are **unifying model** and **frame memory**.

## 2.6 Functions of DBMS

Some important functions of DBMS are as follows:

### 1. Data Processing

The most important function of DBMS is data processing. It includes creation, storage and arrangement of data in database. DBMS also provides access to data stored in databases.

### 2. User-accessible Catalog

A **catalog** is an object contains all information about the database. It includes schema information, user information, user rights etc. The administrative user of database should be able to access catalog. It is an important function of DBMS to provide access to the catalog.

### 3. Transaction Support

A collection of all steps to complete a process is known as **transaction**. DBMS should support transaction. It must ensure that all steps in a trasaction are executed successfully or none of them is executed. This facility ensures that the database is always in consistent state even if a transaction fails due to some problem such as system crash or power failure etc.

### 4. Concurrency Support

A situation in which two or more persons access the same record simultaneously is called **concurrency**. This situation may result in loss of information or loss of integrity. DBMS must provide the facility of concurrency. It enables multiple users to access the same record simultaneously without any loss of data.

### 5. Recovery Services

A DBMS must provide a mechanism to recover a database if it is damaged in any way. It ensures that the database remains in a consistent state. DBMS must also ensure that data loss during recovery process is minimum.

### 6. Data Communication Support

DBMS must provide data communication support in different ways. It must be capable of integrating with communication software. The users usually access the database from workstations. The workstations may be connected to the host computer or located at distant locations. The workstations communicate with host computer over a network. DBMS receives their requests as communication messages. All communication is handled by **Data Communication Manager (DCM)**. The DCM is not part of DBMS but the DBMS should be capable of integrating with DCM for data communication.

### 7. Integrity Services

Integrity means accuracy and reliability of data. It is maintained by applying particular constraints on the data. Data integrity rules ensure the correctness and consistency of stored data. DBMS must provide integrity services to maintain the integrity of data. It must protect the database from false and incorrect data.

### 8. Authorization Services

DBMS must ensure the security of database through authorization services. It should ensure that only authorized users can access the database. The authorization is normally implemented with the help of passwords etc. DBMS must also provide different levels of authorization. For example, one user may be authorized to access full database but another user may be allowed to access only a part of database.

## 2.7 Database Development Process

The development of complete database application is lengthy and complicated process. Different strategies can be used to develop database applications which are as follows:

### 2.7.1 General Strategies

A database application is developed to satisfy the requirements of the user. It is very important to understand these requirements in detail. The application should be developed according to the expectation of user. Different techniques are used to find the requirements and needs of users such as interviews. The requirements should be defined as early as possible in development process. There are two strategies to develop a database application:

1. Top-Down Development
2. Button-up Development

#### 1. Top-Down Development

This strategy starts with general issues and moves to specific issues. First of all, it is important to find out general goals of organization and the means by which these goals can be achieved. The requirements are defined that must be satisfied to reach these goals. This study gives an abstract data model of the system.

The user moves to detailed and specific issues using this model. This process identifies a particular database and related application to be developed. Finally, high-level data model is transformed into low-level models. All identified systems, databases and applications are developed.

#### 2. Bottom-Up Development

This strategy starts with specific issues and moves to general issues. The user begins by identifying a specific system to be developed. The requirements are found by studying the existing system and by interviewing different users.

## 2.8 System Development Life Cycle

System development life cycle is a conventional way to develop an information system. It consists of many steps and involves different persons. The steps of SDLC are as follows:

### 2.8.1 Preliminary Investigation

Preliminary investigation is the first phase of SDLC. Its main objective is to identify deficiencies and requirements in the user's current environment. An important result of the preliminary investigation is whether the system to be developed is feasible or not.

Feasibility is determined on the following parameters:

- Whether current technical resources or technology is available in the developer's organization or in the market that is capable of handling the user's requirements.
- Whether the system is cost effective economically or financially.
- How effectively the user will operate this software once installed.

**Feasibility study report** is produced at the end of this phase. A final acceptance of the proposed system is taken from the user. The next phase is started when the proposed system is accepted.

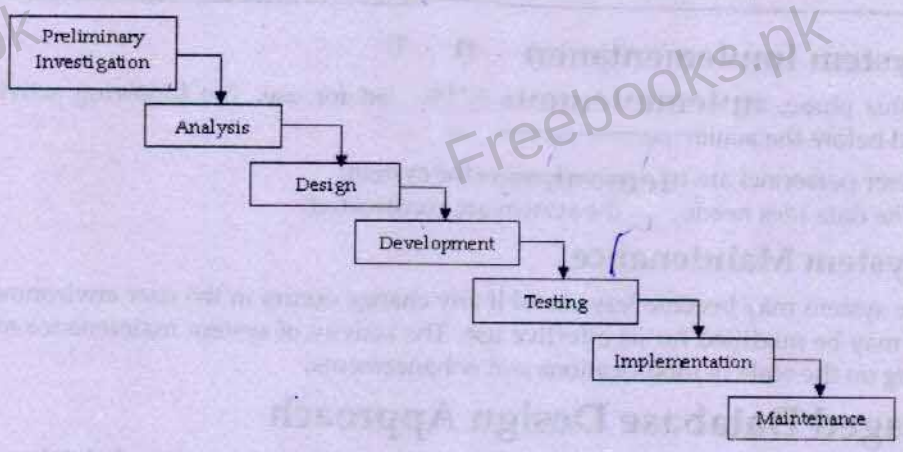



**Figure:** Phases of System Development Life Cycle

## 2.8.2 Requirement Analysis

In this phase, the current business system is studied in detail to find out how it works and where the improvements are required. It includes a detailed study of various operations performed by the system and their relationship within and outside the system. The analyst and user work closely during the complete analysis phase. A detailed document is prepared at the end of this phase called **requirement specifications.**

## 2.8.3 System Design

The requirement analysis phase provides the requirements of the system. The next phase is to design the new system to satisfy these requirements. The design phase states how a system will meet the requirements identified during systems analysis phase as mentioned in the requirements specifications.

Some activities performed during design phase are as follows:

- Identification of data entry forms along with the data elements
- Identification of reports and outputs of the new system
- Design the form or display as expected in the system. This may be done on paper or on a computer display using any design tools.
- Identification of data elements and tables for database creation
- Procedures for deriving the output from given input

The document produced at the end of this activity is called **design specification**. The detailed design specification is given to the programmers to start software development.

## 2.8.4 Software Development

In this phase, actual coding of the programs is done. Programs are tested using dummy data. Programmers also prepare the documentation related to programs. The documentation explains how and why a certain procedure was coded in a specific way.

## 2.8.5 System Testing

After the programs are tested individually, the system is tested as a whole. During system testing phase, all software modules are integrated and tested to ensure that they are running according to the specifications. Special test data is prepared as input for processing. The results are examined to ensure that they are correct.

### 2.8.6 System Implementation

In this phase, the developed system is installed for use. The following activities are performed before the actual usage of the system:

- User personnel are trained to operate the system.
- The data files needed by the system are constructed.

### 2.8.7 System Maintenance

The system may become less useful if any change occurs in the user environment. The software may be modified for its effective use. The activity of system maintenance may vary depending on the scale of modifications and enhancements.

## 2.9 Staged Database Design Approach

Another way to design an information system is known as **staged database design approach**. It is a top-down approach. It begins by analyzing the general requirements of the organization. As the process continues, these problems are analyzed in more details. The steps in this approach are as follows:

### 1. Analyze User Environment

The first step in designing a database is to understand and analyze the current user environment. The designer should closely study the current system and its outputs. He should also interview different users to know their current and future requirements.

### 2. Develop Logical Data Model

The designer develops a logical data model for the organization after analyzing the user environment. This data model consists of all entities, attributes and relationships. The designer also determines the following things:

- Types of applications and transactions
- Types of databases access
- Volume of transactions
- Volume of data
- Frequency of databases access
- Budgetary restrictions
- Performance requirements

### 3. Choose a DBMS

The designer chooses a particular database management system on the basis of logical data model. The selected DBMS should satisfy all requirements and constrains identified in the logical data model.

### 4. Map Logical Model to DBMS

The designer maps the logical data model to available data structure of the selected database management system.

### 5. Develop Physical Model

The designer creates the exact layout of data according the facilities of selected DBMS and available resources of software and hardware.

## 6. Evaluate Physical Model

The designer evaluates the physical model by checking the performance of applications and transactions. The designer may implement a particular portion of database to validate the user views and performance effectively.

## 7. Perform Tuning

Tuning is performed to improve the performance of database. Different modifications are made to the physical model if required.

## 8. Implement Physical Model

The designer implements the physical model if evaluation is satisfying. The database becomes functional.

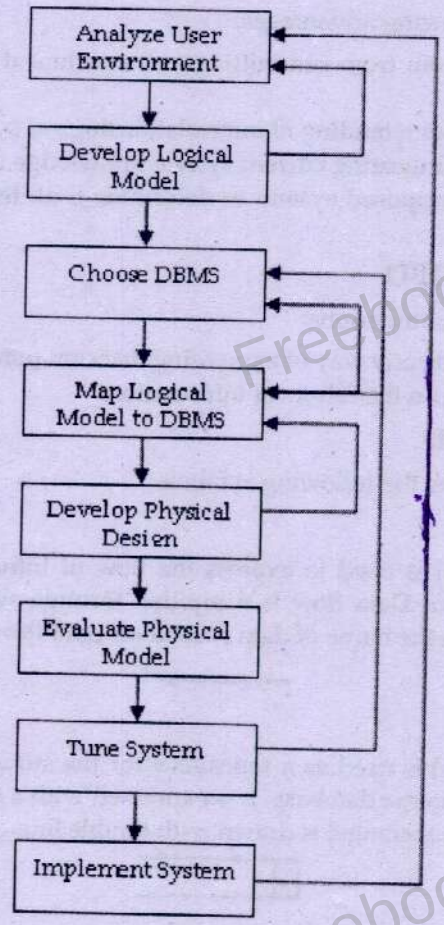



**Figure:** Staged Database Design

The above figure shows that different steps can be repeated at different stages of the development process. For example, the database designer can review the user environment during the development of logical model. While mapping the logical model to physical model, he can change the selection of DBMS. Similarly, if an error occurs and he has to tune the system, he may need to change DBMS or remap the logical model. Even at the last step, he may review all steps from the very beginning.

## 2.10 Design Tools

Design tools are used to describe the design process in a standard way. A standard tool is important because it provides standard notation for desingning specific systems. If there is no standard tool, everyone may use different design notations that can be difficult to understand for others. This situation can result in more critical problem if both persons are working on the same system.

### 2.10.1 Data Flow Diagram

The most commonly used tool to design database system is **data flow diagram (DFD)**. A data flow diagram shows the flow of data through an organization. It is used to design systems graphically. DFD is very simple and it hides complexities of the system.

#### 2.10.1.1 Advantages of DFD

DFD provides the following advantages:

- It provides the freedom from committing to the technical implementation of system too early.
- It helps in further understanding of interrelationships of systems and subsystems.
- It is helpful in communicating current system knowledge to users.
- It is an analysis of proposed system to determine if all the data and processes have been defined.

#### 2.10.1.2 Limitations of DFD

DFD has the following limitations:

- DFD does not provide any way of expressing decision points.
- DFD is only focused on the follow of information.

#### 2.10.1.3 Symbols in DFD

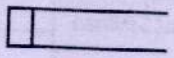
Data flow diagram uses the following symbols:

**Data Flow**

The data flow symbol is used to express the flow of information from one entity to another entity in the system. Data flow is a pipiline through wich packets of information flow. An arrow labeled with the name of data is used for data flow:

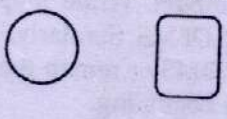
**Data Store**

The data store symbol is used as a repository for the storage of data. It indicates that data is permanently stored in the database. It is expressed with a rectangle that is open on the right. The right width of the rectangel is drawn with double lines.



**Process**

The process symbol is used to express the transformation of incoming data flow into outgoing data fow. It is used to indicate the occurence of an action. It is used when data is transformed from on form to another form. An oval or rounded recntagle is used for this:

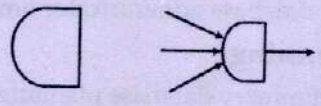


## External Entity

An entity that interacts with the system from outside the system is called **external entity**. The external entities interact with the system in two ways. They may receive the data from the system or may produce data fro the system. It is expressed as rectangle:

## Collector

The collector symbol is used to express several data flow connection terminating at a single location. It is used to show the convergence of data to a single point. The following symbol is used to represent a collector:



## Separator

The separator is used to separate data from a single source to multiple sinks. The following symbol is used to represent a separator:

**Context diagrams** are also used to represent and depict a system. A DFD is a more detailed representation of the system. A context diagram only deals with the boundary of the system. It gives the overall picture of a system. It represents the software model as a single and large process with input and output data. It is displayed by incoming and outgoing arrows respectively. It focuses on the main dataflow in a system. It establishes the relationships between external entities of a system and the system via the data (flows) that they exchange. To enable a DFD to start from a context diagram, single bubble of the context diagram is partitioned into more bubbles to reveal the details of processing in the system.

The dataflow diagrams are basically process-oriented. They are used to model the functions performed by a system.

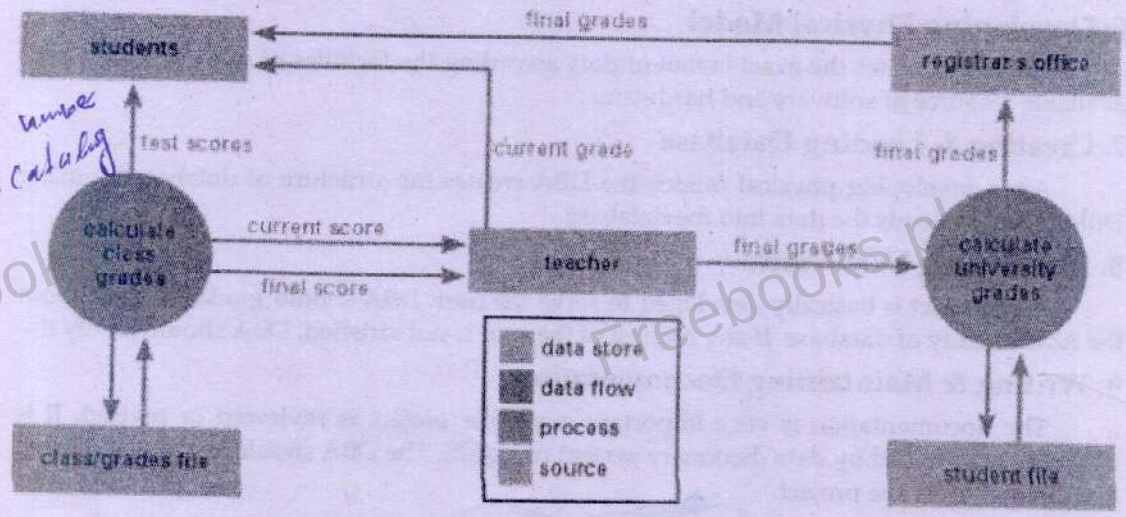



**Figure:** A data flow diagram shows how data moves through the existing system.

## 2.11 Database Administrator (DBA)

DBA is an important person in the development of any information system. He is responsible for design, operation and management of database. He must be technically competent, good manager and skilled diplomat. He should have good communication skills. Managerial skills are important in planning, coordinating and carrying out different tasks. Technical skills are required to understnd the complex hardware and software issues. Diplomatic skills are important in communicating with users, determining their needs and negotiating agreements etc.

### 2.11.1 Functions of DBA

The main responsibilities of database administrator are as follows:

#### 1. Preliminary Database Planning

DBA may participate in preliminary database planning if appointed early.

#### 2. Identifying User Requirements

DBA identifies the current user environment. He closely studies the current system and its outputs. He also interviews different users to know their current and future requirements.

#### 3. Developing & Maintaining Data Dictionary

Data dictionary is a very useful collection of data about database. The DBA stores data item names, sources, meanings and usage in data dictionary. The data dictionary is revised regularly to update it as the project continues.

#### 4. Designing Logical Model

After analyzing the user environment, the DBA develops a logical data model for the organization. This data model consists of all entities, attributes and relationships.

#### 5. Choosing a DBMS

The DBA chooses a particular database management system on the basis of logical data model. The selected DBMS should satisfy all requirements and constrains identified in the logical data model.

#### 6. Developing Physical Model

The DBA creates the exact layout of data according the facilities of selected DBMS and available resource of software and hardware.

#### 7. Creating & Loading Database

After developing physical model, the DBA creates the structure of database by using DBMS. He also loads the data into the database.

#### 8. Developing User Views

Any project is basically developed to serve the user. DBA should guide the user about the functionality of database. If any request of the user is not satisfied, DBA should justify it.

#### 9. Writing & Maintaining Documentation

The documentation is very important when the project is reviewed or revised. It is normally maintained by data dictionary system of DBMS. The DBA should ensure the proper documentation of the project.

### 10. Developing & Enforcing Data Standard

Data in database must be inserted according to the standard required by organization. DBA ensures that the inserted data is always according to these standards. The user interface should guide the user to insert proper data. For example, the text fields may contain default values that make it easy for the user to judge the type of data to be inserted. **Integrity** means that database must always satisfy the rules that apply to the real word. For example, if any employees can only work in one department, database should not allow an employee to be registered in two departments. **Consistency** means that two different pieces of data cannot contradict each other. For example if date of birth of an employee is 7/8/78 at one place and it is 12/12/80 at another place then the database is inconsistent. DBA must ensure that such situation never occurs.

### 11. Developing Operating Procedures

DBA should establish procedures for different operations. The procedures include security and authorization, recording hardware and software failures, taking performance measurements, shutting down database property, restarting and recovering after failure.

### 12. Training the Users

The DBA should train end users, application programmers and other users so that they may use the database effectively.

### 13. Helping Database Users

The database administrator helps the database users by:

- Making sure that the data they require is available
- Assisting them on using correctly the system

### 14. Defining Backup and Recovery Procedures

If any portion of database is damaged by human error or by hardware, it should be restored as soon as possible. It is important to backup the information of the database on a backup server so that it may be recovered in case of emergency.

### 15. Monitoring Performance

The DBA should constantly monitor the performance of the database. He should respond to the complaints or suggestions from the users. The DBA should make sure the database should always work in its optimum performance. If there is any problem in the functioning of the database, DBA should solve the problem by taking appropriate measure.

### 16. Tuning & Reorganizing

If the performance of the database is disturbed due to huge amount of data, the DBA should tune it. He should add or change the indexes. In some cases, the DBA may have to change the physical model and reload the database.

## 2.12 Data Administrator (DA)

The need of data administrator arises in a very large organization where many databases may exist. Data administrator is responsible for the whole information resource. He develops the requirements for database, develops logical design and other non-technical functions. He also controls and manages database, establishes data standards, communicates with users, prepares logical design, develops data dictionary, plans the development of database and application programs, trains users and maintains documentation.

## 2.13 Data Dictionary

**Data dictionary** is a repository of information that describes the logical structure of the database. It contains record types, data item types and data aggregates etc. Data dictionaries in some systems store database schema and can be used to create and process database. Data dictionary contains **metadata**. Metadata is the data about the data stored in the database.

### 2.13.1 Uses of Data Dictionary

Different uses of data dictionary are as follows:

- **Information about Data:** It is used to collect and store information about data in a central location. It helps the management to get control over data as a resource.
- **Communication with Users:** It provides great help in communication as it stores exact meanings of data items. An exact definition of each item should be stored in data dictionary that can be used in case of any problem.
- **Record of Change in Database Structure:** It keeps track of changes to the database structure. The changes such as creation of new data item or modification of data item descriptions should be record in data dictionary.
- **Determining the Impact of Change:** Data dictionary records each item and its relationships. DBA can see the effects of a change.
- **Recording Access Control Information:** Data dictionary stores all information about different authorized users. It also contains the types of access for all users.
- **Audit Information:** It can also keep record of each access to the database. This information can later be used for audit purposes.

### 2.13.2 Types of Data Dictionaries

Different types data dictionaries are as follows:

#### 1. Integrated Data Dictionary

A data dictionary that is part of DBMS is called **integrated data dictionary**. It performs many functions throughout the life of the database not only in design phase. There are two types of integrated data dictionary:

- **Active Data Dictionary:** The integrated data dictionary is called **active** if it is checked by DBMS every time a database is accessed. It is always consistent with actual database structure. It is automatically maintained by the system.
- **Passive Data Dictionary:** The integrated data dictionary is called **passive** if it is not used in day-to-day database processing.

#### 2. Freestanding Data Dictionary

A data dictionary that is available without a particular DBMS is called **freestanding data dictionary**. It can be a commercial product or a simple file developed by the designer. Many CASE packages provide a data dictionary tool. It is preferable in initial design stages before choosing any particular DBMS.

## 2.14 Logical Database Design

The **logical** database design contains the definition of the data to be stored in database. It also contains the rules and information about the structure and type of data. All entities, their attributes and their relationships are described in logical model. It is the complete description of data stored in database.

## 2.14.1 Logical Database Design Process

An overview of logical database design process is as follows:

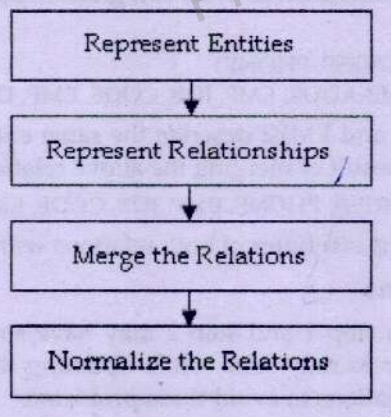



**Figure:** Logical Design Process

### 1. Represent Entities

Each entity in a E-R diagram is represented as a relation in relational model. In this process, the name of entity becomes the name of relation. The identifier of entity type becomes the primary key of relation. The remaining attributes of the entity type become non-key attributes of the relation.

Following example explains the process of converting an entity into a relation:

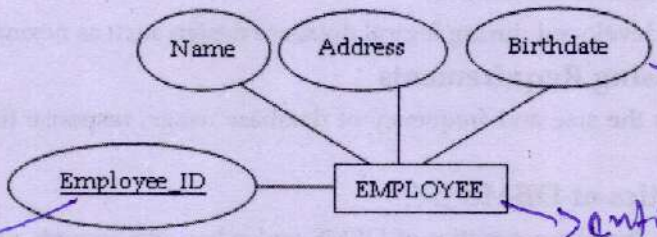



**Figure:** EMPLOYEE entity

| (EmployeeID | Name | Address | Birthdate |
|---|---|---|---|

**Figure:** EMPLOYEE Relation

In the above example, EMPLOYEE entity is converted into relation. The attributes of the entity are fields of the relation. The data model describes EmployeeID as an identifier and is underlined. The above relation is using EmployeeID as primary key for the relation.

### 2. Represent Relationships

Each relationship in an E-R diagram must also be represented in relational model. The representation depends on the nature of relationship. In some cases, a relationship is represented by making the primary key of one relation a foreign key of another relation. In some cases, a separate relation is created to represent a relationship.

### 3. Merge the Relations

In some cases, there may be **redundant relations**. It means that two or more relations may describe the same entity type. The redundant relations must be merged to remove the redundancy. This process is also known as **view integration**. Suppose there are two relations as follows:

**EMP1** (EMPNO, NAME, ADDRESS, PHONE)
**EMP2** (EMPNO, ENAME, EMP-ADDR, EMP_JOB_CODE, EMP_DOB)

The above tables EMP1 and EMP2 describe the same entity EMPLOYEE. They can be merged into one relation. The result of merging the above relations is as follows:

**EMP** (EMPNO, NAME, ADDRESS, PHONE, EMP_JOB_CODE, EMP_DOB)

The new relation contains attributes of both relations without any repeating attributes.

### 4. Normalize the Relations

The relations created in step 1 and step 2 may have some unnecessary redundancy. Some certain anomalies or errors may arise while updating these relations. The process of **normalization** refines these relations to avoid these problems.

## 2.15 Physical Database Design

**Physical design** is the last stage of database design process. The major objective of physical dataase design is to implement the databse as a set of records, files, indexes and other data structures.

### 2.15.1 Major Inputs to Database Design

Three major inputs to physical database design are as follows:

### 1. Logical Database Structure

These are developed during logical database design such as normalized relations.

### 2. User Processing Requirements

It includes the size and frequency of database usage, response time, security, backup and recovery etc.

### 3. Characteristics of DBMS

It includes the characteristics of DBMS and other components of computer operating environment.

### 2.15.2 Components of Physical Database Design

Different components of physical database desing are as follows:

### 1. Data Volume and Usage Analysis

It is used to estimate the size or volume and usage patterns of database. The estimate of database size is used to select the physical storage devices. It is also used to determine the costs of storage. The estimate of usage patterns are used to select file organization and access methods. It is also used to plan for the use of indexes and a strategy for data distribution.

### 2. Data Distribution Strategy

Many organizations are using distributed computer networks now a days. These organization face a significant problem in physical database design. The problem is that they have to decide and select nodes or sites in network at which data will be located physically.

The basic data distribution strategies are as follows:

### i. Centralized

In this strategy, all data is located at a single site. It is simple and easy to conduct. This strategy has three disadvantages:

- Data stored at remote sites is not accessible readily.
- Data communication costs may be very high.
- The database system fails totally when the central system fails.

### ii. Partitioned

In this strategy, the database is divided into partitions or fragments. Each partition is assigned to a particular site. The major advantage of this strategy is that data is moved closer to local user. Data becomes more easily accessible.

### iii. Replicated

In this strategy, the full copy of the database is assigned to more than one site in the network. This strategy maximizes local access. But it creates update probelms because each database change must be reliably processed and synchronized at all sites.

### iv. Hybrid

In this strategy, the database is divided into critical and non-critical fragments. The critical fragments are stored at multiple sites. The non-critical fragments are stored at one site only.

## 3. File Organization

File organization is a technique for physically arranging the records of a file on secondary devices. The system designer must recognize several constraints for selecting a file organization. These constrains inlcude the following:

- Physical characteristics of secondary storage devices
- Available operating systems and file management software
- User requirements for storing and accessing data

## Criteria to Select File Organization

The criteria for selecting a file organization are as follows:

- Fast access for data retrieval
- High throughput for processing transactions
- Efficient use of storage space
- Protection from failure or data loss
- Minimizing need for data re-organization
- Security from unauthorized use

## File Organization Methods

The files are organized on storage media in the following methods:

### a. Sequential Files

The records in **sequential file organization** are stored in sequence. A sequence means the records are stored one after the other. The records can be retrieved only in the sequence in which they were stored. The principal storage media for sequential files is magnetic tape.

The major disadvantage of sequential access is that it is very slow. If the the last record is to be retrieved, all preceding record are read before reaching the last record.

### b. Direct or Random Files

The records in **direct file organization** are not stored in a particular sequence. A key value of a record is used to determine the location to store the record. Each record is accessed directly without going through the preceding records.

This file organization is suitable for storing data on disk. Direct file organization is much faster than sequential file organization for finding a specific record.

A problem may occur in this type of files known as **synonym**. The problem occurs if the same address is calculated to store two or more records.

### c. Indexed Sequential Files

In **indexed sequential file organization**, records are stored in ascending or descending order. The order is based on a value called **key**. Additionally, indexed file organization maintains an **index** in a file.

An **index** consists of key values and the corresponding disk address for each record in the file. Index refers to the place on a disk where a record is stored. The index file is updated whenever a record is added or deleted from the file.

The records in indexed file organization can be accessed in sequential access as well as **random access** or **direct access**. The records in this file type require more space on storage media. This method is slower than direct file organization as it requires to perform an index search.

## 4. Indexes

An **index** is a table that is used to determine the location of rows in a table. Indexes are used to speed up the sorting and searching process. The performance of database is improved with these indexes. The index may be created on primary key, secondary key and foreign key etc.

## 5. Integrity Constraints

**Database integrity** means the correctness and consistency of data. It is another form of database protection. Integrity is related to the quality of data. Integrity is maintained with the help of **integrity constraints**. These constraints are the rules that are designed to keep data consistent and correct. They act like a check on the incoming data. It is very important that a database maintains the quality of the data stored in it. DBMS provides several mechanisms to enforce integrity of the data.

## Short Questions

### Q.1. What is three-level architecture?

Three-level architecture is the basis of modern database architecture. Database can be viewed at three levels. The three levels are depicted by three models known as three-level schema. The purpopse of three-level architecture is to separate the way the database is physically repsesented from the way the user thinks about it.

### Q.2. Name and explain the purpose of three schemas in ANSI/SPARC three schema model.

**External Schema:** The external schema or user view is a representation of how users view the database. An external schema portrays just a portion of the database.

**Conceptual Schema:** A conceptual schema is a complete logical view of database that contains a description of all data and relationships in database. The conceptual schema is logical. It is independent of any particular means of storing data.

**Internal Schema:** An internal schema is a representation of a conceptual schema as physically stored using a particular product and/or technique. The description of a set of tables, keys, foreign keys, indexes and other physical structures is an internal schema.

### Q.3. How are external schemas used to devise one conceptual schema?

External schemas are descriptions of the world as end users see the world. During the database design process, designers collect information from end users and develop different external schemas for different end users. These external schemas are then combined into one conceptual schema.

### Q.4. What is hierarchical database? Write down the advantages and disadvantages of Hierarchical database model.

Hierarchical database organize data in a series like a family tree or organizational chart. The hierarchical database has branches made up of parent and child records. Each parent record can have multiple child records but a child record can have only one parent.

**Advantages**

1. Hierarchical DBMS is good for one-to-many relationships.
2. It can store large numbers of segments and process information efficiently.
3. It is a simple structure and minimizes disk input and output.

**Disadvantages**

1. It is less user-friendly.
2. It creates inflexibility and programming complexity.
3. It is difficult ot reorganize the database as hierarchy has to be maintained.

### Q.5. What is a data model? List main types of data models.

A collection of concepts to describe and manipulate data, relationships between data and constraints on data is called data model. Object-based data models such as the Entity–Relationship model. Record-based data models such as the relational data model, network data model, and hierarchical data model. Physical data models describe how data is stored in the computer .

### Q.6. What is relational database? List its advantages and disadvantages.

Relation database organizes data in two-dimensional tables called relation. Each row in a table is called tuple and each column is called an attribute. Tuple is also called record or row. The ranges of values that an attribute can have is called domain.

**Advantages**

1. It has the capacity to link multiple files together.
2. It provides the facility of data independence.
3. It provides the facility of structural independence.

4. It provides the facility of data integrity.
5. It has the capability to add new fields.
6. It provides the facility to establish relationship at any time.

**Disadvantages**

1. It is a bit poor in processing efficiency.
2. Response time can be very slow if large numbers of accesses to data are required to select, join, and extract data from tables.

**Q.7. Describe the role and responsibilities of a database administrator (DBA).**

1. The development and management of the organization's databases.
2. He must work with a number of people including systems analysts, programmers, users, managers, and security personnel.
3. He assists in design and implementation of database.
4. He helps to establish policies and procedures
5. He assists to ensure security of the database.

**Q.8. What is difference between Data Administrator and Database Administrator.**

Data administrator is responsible for corporate data resource. It includes non-computerized data. He manages shared data of users or application areas of an organization. He determines long-term goals and enforces standards, policies and procedures. He also determines data requirements and develops conceptual and logical desing. His work is DBMS-independent.

Database adminstrator is more technically oriented. He requires the knowledge of a DBMS. He develops and maintains systems using DBMS. He executes plans to achieve goals. He enforces standards, policies and procedures developed by DA. He develops logical and physical desing. His work is DBMS-dependent.

**Q.9. What is a data dictionary?**

Data dictionary is a repository of information that describes the logical structure of the database. It contains record types, data item types and data aggregates etc. The data dictionaries in some systems store database schema and can be used to create and process the database. Data dictionary contains metadata. Metadata is the data about the data stored in the database.

**Q.10. List some uses of data dictionary.**

1. It is used to collect and store information about data in a central location.
2. It provides great help in communication as it stores exact meanings of data items.
3. It keeps track of changes to the database structure.
4. It records each item, its relationships and users. DBA can see the effects of a change.
5. It stores all information about different authorized users.
6. It can also keep record of each access to the database to be used for audit purposes.

**Q.11. Describe different types of data dictionaries.**

Different types data dictionaries are as follows:

**1. Integrated Data Dictionary:** A data dictionary that is part of DBMS is called integrated data dictionary. They perform many functions throughout the life of the database not only in design phase.

**2. Freestanding Data Dictionary:** A data dictionary that is available without a particular DBMS, it is called freestanding data dictionary. It can be a commercial product or a simple file developed by the designer. Many CASE packages provide a data dictionary tool. It is preferable in initial design stages before choosing any particular DBMS.

**Q.12. Differentiate between active and passive data dictionary.**

The integrated data dictionary is called active if it is checked by DBMS every time a database is accessed. It is always consistent with actual database structure. It is automatically maintained by the system. The integrated data dictionary is called passive if it is not used in day-to-day database processing.

**Q.13. What are the effectiveness measures of database design?**

The major effectiveness measures of database design are as follows:

a. Ensuring that data can be shared among users for a variety of applications.

b. Maintaining data that are both accurate and consistent.

c. Ensuring all data required for current and future applications will be readily available.

d. Allowing the database to evolve and the needs of the user to grow.

e. Allowing users to construct personal view of data without concerning the way it is physically stored.

**Q.14. What elements of a data flow diagram should be analyzed as part of data modeling?**

Data stores, data flows and even processes provide information for data modeling. A data store often represents one or more data entities and their associated attributes. All data in data flows must either be stored in some entity, be computed from data in entities or pass through the system. The description of a process can indicate a business rules that must be represented in the data model

---

## Multiple Choice

1. **Which of the following is not one of the three schemas used in the ANSI/SPARC**
   a. External b. Internal c. Implementation d. Conceptual
2. **Which of three schemas used in three-schema model represents how users view database?**
   a. External b. Internal c. Implementation d. Conceptual
3. **Which of three schemas used in three-schema model is a complete logical view of database?**
   a. External b. Internal c. Implementation d. Conceptual
4. **Which of three schemas used in three-schema model shows physical storage using a particular product or technique?**
   a. External b. Internal c. Implementation d. Conceptual
5. **When discussing a data model under construction, users speak in terms of _____ schema they work with, which database developers must translate into _____ schema.**
   a. External, internal b. External, conceptual
   c. Internal, implementation d. Internal, conceptual
6. **Although all database professionals are concerned with performance, select the database professional that is most likely to be concern database performance:**
   a. Database administrator b. Application developer
   c. Website designer d. Database designer
7. **Which data model creates parent-child relationships between data elements and restricts each child to have just one parent?**
   a. Hierarchical b. Network c. Relational d. Object
8. **Which data model creates parent-child relationships between data elements and enables each child to have more than one parent?**
   a. Hierarchical b. Network c. Relational d. Object
9. **Which of the following data models stores data in table structures and performs various operations on table rows and columns?**
   a. Hierarchical b. Network c. Relational d. Object

**10. The older logical database model that organizes data in a treelike structure is:**

a. Hierarchical b. Network c. Relational d. Object

**11. The hierarchical database is very efficient when:**

a. Handling large amounts of data. b. Handling little amounts of transactions
c. Handling many transactions d. a and c

**12. Which of the following is NOT a major area for the database administrator?**

a. Planning b. Programming c. Implementation d. Security

**13. DBA stands for:**

a. Database Administrator b. data basic Administration
c. Database Application d. Database authority

**14. Which of the following represents a collection of concepts that are used to describe the structure of a database**

a. Data warehouse b. Data model c. Data structure d.Data type

**15. The relational model_____.**

a. was first proposed in 1970 b. was developed by E. F. Codd
c. was developed at IBM d. All of these

**16. Physical database design decisions must be made carefully because of impacts on:**

a. Data accessibility b. Response times c. Security d. All

**17. An advantage of partitioning is:**

a. Efficiency b. Extra space and update time c. Both A and B d. None

**18. A disadvantage of partitioning is:**

a. Simplicity b. Extra space and update time c. Both a and b d. None

**19. Merge relation is important because:**

a. Different views may need to be integrated.
b. New data requirements may produce new relations to be merged.
c. Both a and b
d. None

**20. Organizing the database in computer disk storage is done in:**

a. Logical design b. Physical designc. Analysis d. Implementation

**21. Which of the following are basic data distribution strategies?**

a. Centralized b. Partitioned c. Replication d. All

**22. _____ refers to a method of database distribution in which different portions of the database reside at different nodes in the network.**

a. Splitting b. Partitioning c. Replication d. Dividing

**23. _____ refers to a method of database distribution in which one database contains data that are included in another database.**

a. Splitting b. Partitioning c. Replication d. Dividing

## Answers

| | | | |
|---|---|---|---|
| 1. c | 2. a | 3. d | 4. b |
| 5. b | 6. a | 7. a | 8. b |
| 9. c | 10. a | 11. d | 12. b |
| 13. a | 14. b | 15. d | 16. d |
| 17. a | 18. b | 19. c | 20. b |
| 21. d | 22. b | 23. c | |

## True / False

1. The most popular type of DBMS today for PCs as well as for larger computers and mainframes is the hierarchical DBMS.
2. The strength of the relational model is that data in any file or table can be related to another file or table as long as both tables share a common key field.
3. In a hierarchical DBMS, data elements within each record are organized into pieces called segments.
4. In a hierarchical DBMS, an upper segment is connected logically to a lower segment in a parent-child relationship.
5. In a hierarchical DBMS, a parent can have only one child.
6. Network database management systems are no longer used for building new database applications.
7. Many large legacy systems requiring intensive high-volume transaction processing use hierarchical DBMS.
8. Many applications today require databases that can store and retrieve multimedia.
9. The top-level segment in each record of a hierarchical database is called the root.
10. A database requires organizational and conceptual change.
11. Data administration is a very important organizational function, and is fairly easy to implement.
12. The DBA is responsible for the management and development of the organization's database.
13. To ensure proper security features in a database, the DBA should coordinate with organizational managers.
14. The process of organizing and structuring databases efficiently is called database design.
15. Data dictionary is the term that refers to the definition of data stored within the database and controlled by the database management system.
16. Database planning involves working with business area managers to define the firm's data needs.
17. The relational model was first proposed in 1970 by E. F. Codd at IBM.
18. Examining existing forms and reports is part of determining system requirements.
19. Team reviews and user reviews are part of determining system requirements.

## Answers

| | | | |
|---|---|---|---|
| 1. F | 2. F | 3. T | 4. T |
| 5. F | 6. T | 7. T | 8. T |
| 9. T | 10. T | 11. F | 12. T |
| 13. T | 14. T | 15. T | 16. T |
| 17. T | 18. T | 19. F | |

CHAPTER 3

# Entity Relationship Model

## Chapter Overview

E-R Project 4
E-R Project 5
E-R Project 6
E-R Project 7
E-R Project 8
E-R Project 9
E-R Project 10

**Short Questions**
**Multiple Choice Questions**
**True / False Questions**

# 3.1 Entity Relationship Model

Entity-Relationship model is a logical representation of data in an organization. It views the entire system as a collection of entities related to one another. It is used to describe the elements of a system and their relationships. It was introduced by Peter Chen in 1976.

## 3.1.1 Advantages of E-R Model

Some important advantages of E-R model are as follows:

### 1. Conceptual Simplicity

E-R model represents the conceptual of a database along with its entities and relationships in an easy way. It becomes even easier to create and manage the complex database desings by using E-R model.

### 2. Visual Representation

E-R model provides a visual representation of data and the relationships among data. It enables database designers, programmers and end users to understand database easily.

### 3. Effective Communication Tool

The database designer can use E-R model to get different views of data as seen by programmers, managers and end users etc. E-R model works as an effective communication tool to integrate these views.

### 4. Integrated with Relational Database Model

E-R model is well integrated with the relational database model. This integration makes relational database design a very structured process.

# 3.2 Elements of E-R Model

Different elements of an E-R model are as follows:

## 3.2.1 Entities

An entity is a person, place, thing or event for which data is collected and maintained. For example, a library system may contain data about different entities like BOOK and MEMBER. A college system may include entities like STUDENT, TEACHER, and CLASS etc.

Some examples of entities are as follows:

- **Person:** TEACHER, PLAYER, DOCTOR
- **Place:** COUNTRY, CITY
- **Object:** VEHICLE, TOY, FURNITURE
- **Event:** PURCHASE, ADMISSION, REGISTRATION
- **Concept:** ACCOUNT, PROGRAMMING

An entity is represented by a rectangle. The name of the entity is written inside the rectangle. The entity is used in three different meanings that are as follows:

### 3.2.1.1 Entity Type

A set of entities with same attributes is called entity type. All entities in an entity type share common characteristics. It is also known as **entity class**. For example, STUDENT entity class is a set of all students. Similarly, BOOK entity type is a collection of all books etc.

### 3.2.1.2 Entity Instance

A member of an entity class is also known as an entity instance. It is also known as **entity occurrence**. Each entity instance of an entity type has its own value for each instance. For example, a student Abdullah of STUDENT entity type is an entity instance.

### 3.2.1.3 Entity Set

A set of all entities of a particular entity type in the database at a given point of time is called an entity set. For example, an entity set Student may consist of all students in the university. Another entity set Teacher may consist of all teachers in the university etc.

The same name is usually used for both entity type and entity set. For example, BOOK refers to both a entity type as well as the current set of all books in the database.

## 3.2.2 Attributes

The characteristics of an entity are called **attributes** or **properties**. For example, Name, Address, Class and Email of a student are his attributes. All instances of a particular entity class have same attributes. For example, all students of STUDENT entity class have the attributes of Name, Address, Class and Email.

### 3.2.2.1 Attribute Domain

An **attribute domain** is a set of possible values for an attribute. All attributes have domain. The domains may consist of a range of values or some discret values. For example, the domainf for Grade Point Average (GPA) can be from 0 to 4. Similarly, the domain for Gender attribute can be either Male or Female.

The association of a domian with an attribute ensures the integrity of database. The domain is normally defined in form of data type and some additional constaints like range constraint. For example, if the data type of a field is integer, it can store only integer values.

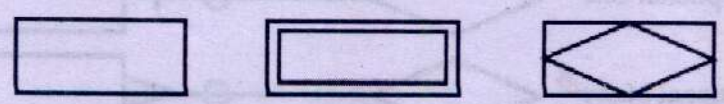
## 3.2.3 Relationships

A **relationship** is a logical connection between different entities. The entities that participate in a relationship are called **participants**. The relationship may be between different entities or between an entity and itself. A relationship is established on the basis of interaction among these entities. For example, a relationship exists between a STUDENT and TEACHER because the teacher *teaches* the students.

A relationship is called **total** if all entities of that entity set may be participant in the relationship. A relationship is called **partial** if some of the entities of that entity set may be participant in the relationship. Suppose a relationship SUPP_PART exists between Supplier and Part. The relationship is total if every part is supplied by a supplier. The relationship is partial if certain parts are available without a supplier.

# 3.3 E-R Diagram

E-R diagram is a graphical representation of E-R model using a set of standard symbols. Different symbols sued in E-R diagram are as follows:



~~String Entity~~ Weak Entity Associative Entity

Strong entt

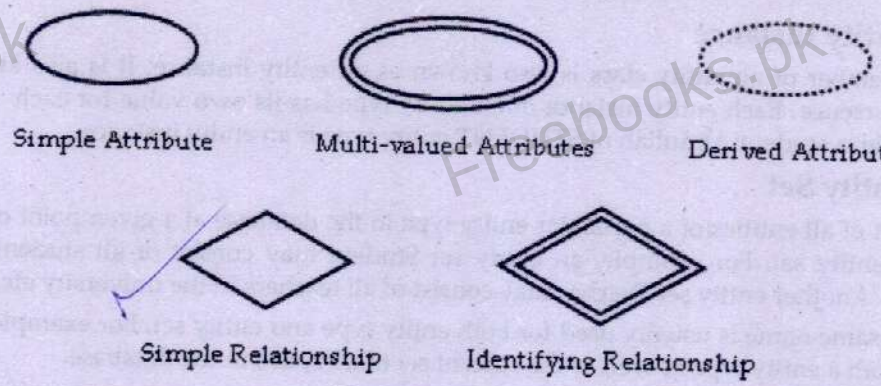



**Figure:** Basic symbols

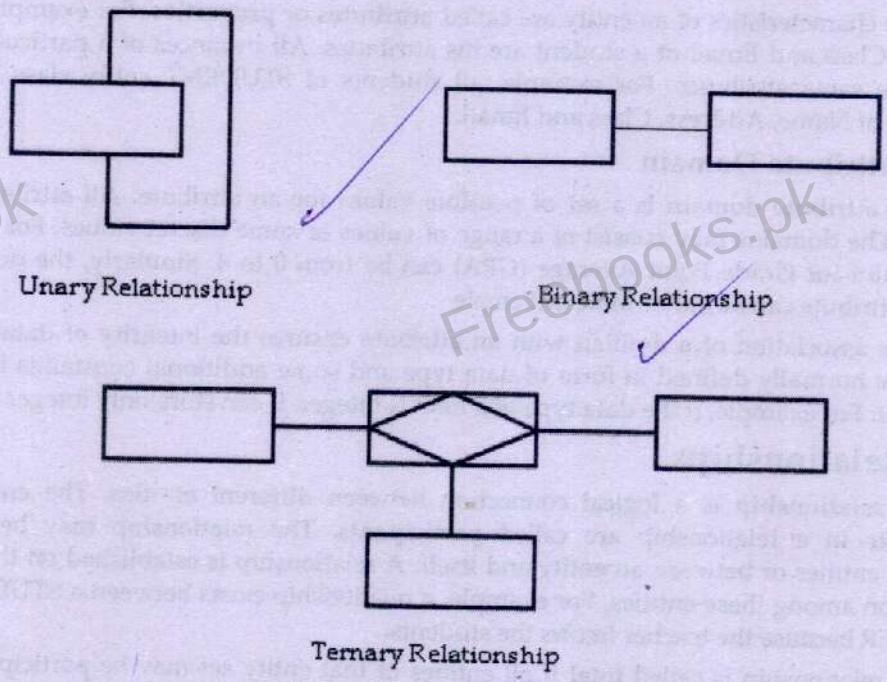



**Figure:** Relationship degrees

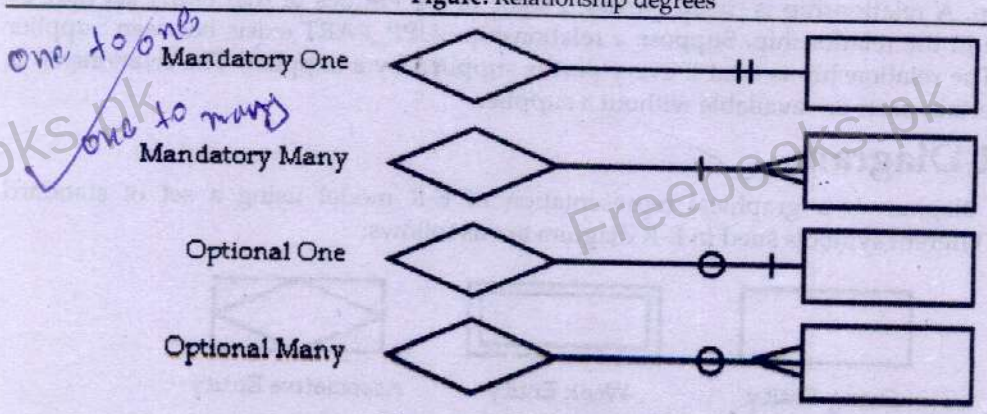



**Figure:** Relationship cardinalities

## 3.3.1 Types of Attributes

Different types of attributes are represented in E-R diagram as follows:

### 3.3.1.1 Simple Attribute

An attribute that cannot be subdivided into smaller component is known as **simple attribute**. It is also called **atomic attribute**. For example, a person can have only one gender and one date of birth.

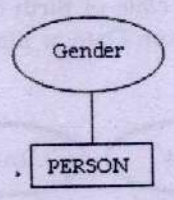



**Figure:** Simple attribute of PERSON entity

### 3.3.1.2 Composite Attribute

An attribute that can be divided into smaller components is called **composite attribute**. For example, Address is an example composite attribute. It can be subdivided into Street, City, Country. Similarly, a Phone Number can be subdivided into Area Code and Number.

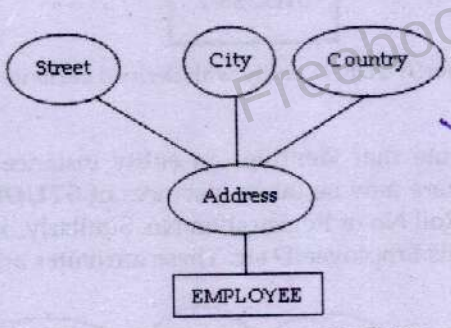



**Figure:** Composite attribute of EMPLOYEE entity

### 3.3.1.3 Single-Valued Attribute

An attribute that may contain single value is called **single-valued attribute**. For example, Age of a person is single-valued attribute. Gender is also a single-valued attribute.

### 3.3.1.4 Multi-Valued Attribute

An attribute that may contain two or more values is called **multi-valued attribute**. For example, a person can have two or more college degrees. Similarly, an employee may have many Skills. Multi-valued attribute is represented by double-line oval.

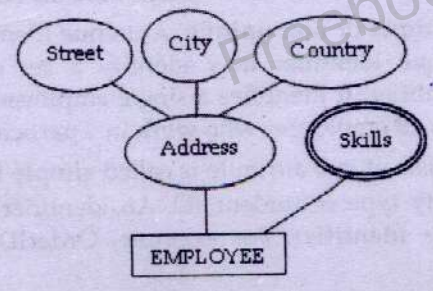



**Figure:** EMPLOYEE entity with multi-valued attribute Skills

### 3.3.1.5 Stored Attributes

An attribute that is stored in a database is called **stored attribute**. Most of the attributes are stored attributes. These are stored and accessed from databases.

### 3.3.1.6 Derived Attributes

An attribute that is not stored in database but derived from another value is called **derived attribute**. the other value can be stored in the database or obtained in some other way. For example, Roll No, Name and Date of Birth of a student can be stored in database. The Age of a student can be derived from Date of Birth. A derived attribute is indicated by using an ellipse with a dashed line.

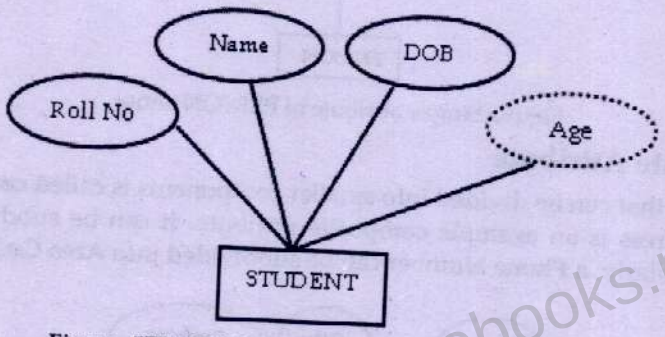



**Figure:** STUDENT entity with derived attribute Age

### 3.3.1.7 Identifiers

Identifier is an attribute that identifies an entity instance among other instances in entity class. For example, there may be many instances of **STUDENT** entity class. But each student is identified by his Roll No or Registration No. Similarly, an employee in EMPLOYEE entity class is identified by his EmployeeID etc. These attributes are known as **identifiers**.

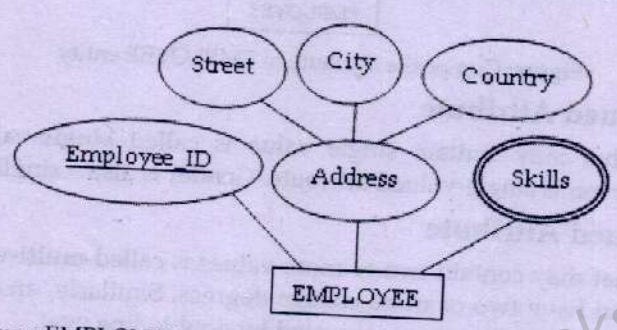



**Figure:** EMPLOYEE entity with identifier attribute Employee_ID

An identifier can be **unique** or **non-unique**. A unique identifier may identify only one entity instance. A non-unique identifier may identify a set of instances. For example, EmployeeID is a unique identifier. It identifies a single employee. But Department is a non-unique identifier. It identifies all employees who work in a particular department.

An identifier that consists of one attribute is called **simple identifier**. For example, the identifier for STUDENT entity type is Student_ID. An identifier that consists of composite attribute is called **composite identifier**. For example, OrderID identifier may consist of OrderNo and Date.

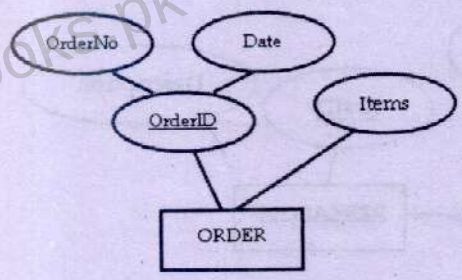



**Figure:** Composite Identifier OrderID

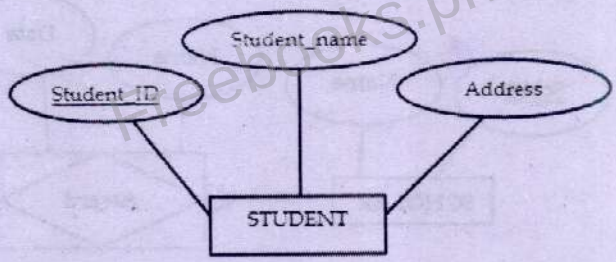



**Figure:** Simple Identifier StudentID

simple identifier student ID

## 3.3.2 Entities

Different types of entities are represented in E-R diagram as follows:

### 3.3.2.1 Weak Entities & Strong Entities

An entity that can exist only if another entity exists is known as **weak entity**. It means that weak entities depend upon the existence of another entity. Suppose we want to store the data of a student after assigning a class to him. It means that the data cannot be stored if CLASS entity does not exist. In order to store the record of the student, we first need to create an entity that represents a class. Here, STUDENT is a weak entity because it depends upon CLASS entity.

The entity on which the weak entity depends is called **owner** or **identifying owner**. A weak entity is represented by double-line square.

An entity that can exist without depending upon the existence of another entity is known as **strong entity**. In the previous example, CLASS is strong entity. A strong entity is also called **parent**, **owner** or **dominant entity**. A weak entity is also called **child**, **dependent** or **subordinate entity**.

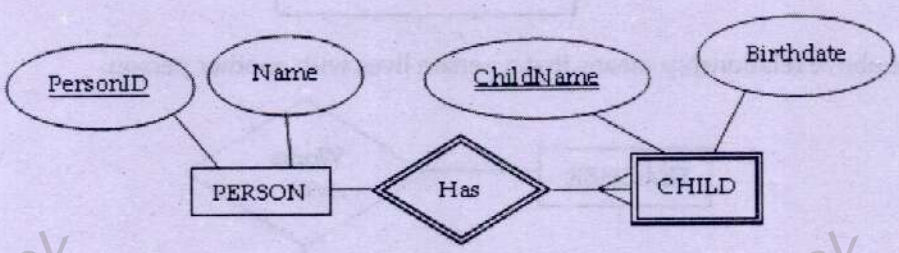



**Figure:** Strong entity PERSON and Weak entity CHILD

In the above figure, CHILD is a weak entity and PERSON is its identifying owner. The relationship of these entities is represented by double-lined diamond. It is called **identifying relationship**.

### 3.3.2.2 Associative Entities

Associative entity is a type of entity that associates the instances of one or many entity types with one another. The attributes of associative represent the relationship between the entity instances. The following figure shows that a scholar gets an award on his research. The entity AWARD has independent meaning to the user.

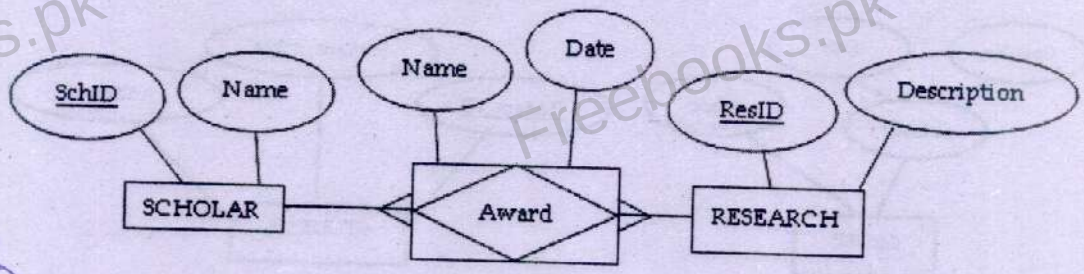



**Figure:** Associative entity Award

## 3.4 Degree of Relationships

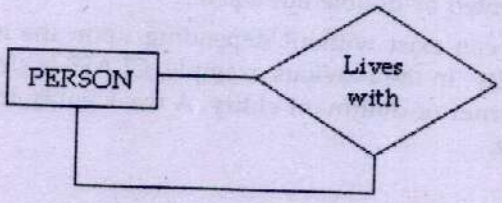
A relationship may consist of many entities. The number of entities in a relationship is called **degree** of relationship. The relationships of degree 2 are most common and are also called **binary relationships**. The types of relationships with respect degree are as follows:

- Unary Relationship
- Binary Relationship
- Ternary Relationship

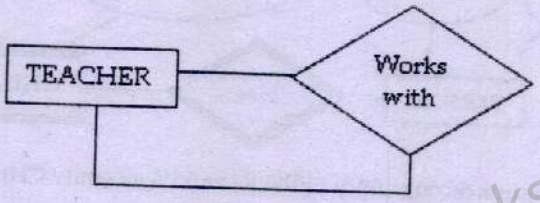
### 3.4.1 Unary Relationship

**Unary relationship** is a type of relationship that is established between the instances of same entity type. This is also known as **recursive relationship**.

**Examples**




The above relationship means that a person lives with another person.




The above relationship means that a teacher works with another teach.

### 3.4.2 Binary Relationships

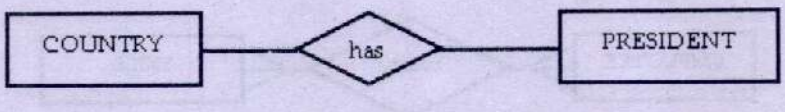
Binary relationships exist between the instances of two entity types. Different types of relationships are as follows:

#### 3.4.2.1 One-to-One Relationship

This type of relationship is used when "for each instance in first entity class, there is only one instance in the second entity class and for each instance in second entity class, there is only one instance in the first entity class".

### Examples


The above relationship means that one chairman runs only one department. Similarly, one department is run by one chairman.



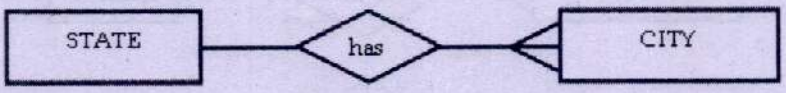

The above relationship means that one country has one president. Similarly, one president is of only one country.


The above relationship means that one manager manages one branch of a bank. Similarly, one branch of a bank is managed by one manager.

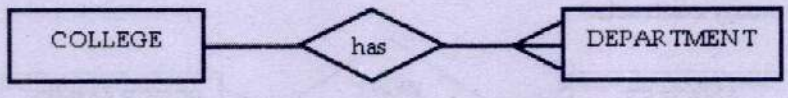
### 3.4.2.2 One-to-Many Relationship

This type of relationship is used when "for each instance in first entity class, there can be many instances in the second entity class and for each instance in second entity class, there is only one instance in the first entity class".

### Examples




The above relationship means that one state can have many cities but one city is belonged to one state.



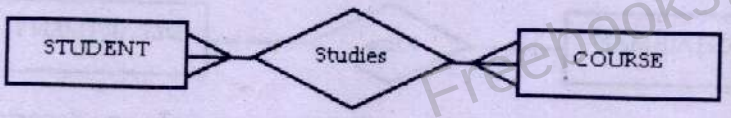

The above relationship means that one college can have many departments but one department is part of only one college.


The above relationship means that one employer can employ many employees but one employee is employed by only one employer.

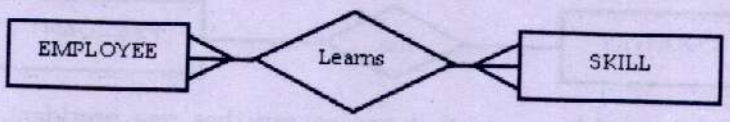
### 3.4.2.3 Many-to-Many Relationship

This type of relationship is used when "for each instance in first entity class, there can be many instances in the second entity class and for each instance in second entity class, there can be many instances in the first entity class".

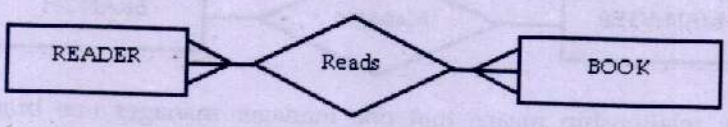
**Examples**




The above relationship means that one student may study many courses and one course may be studied by many students



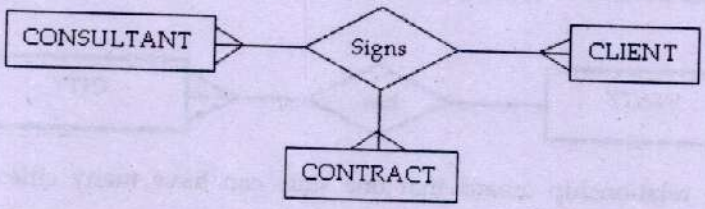

The above relationship means that one employee can learn many skills and one skill can be learned by many employees.



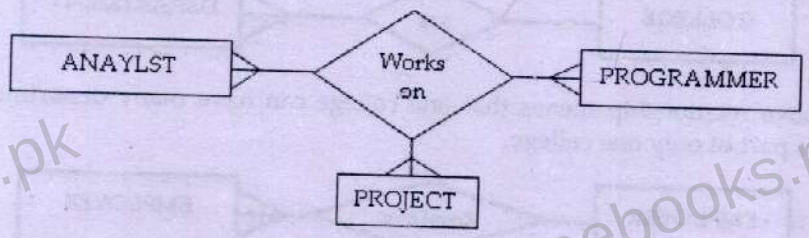

The above relationship means that one reader may read many books and one book may be read by many readers.

## 3.4.3 Ternary Relationship

Ternary relationship exists among the instances of three entity types.




The above relationship means that one or many consultants with one or many clients sign on one or many contracts.



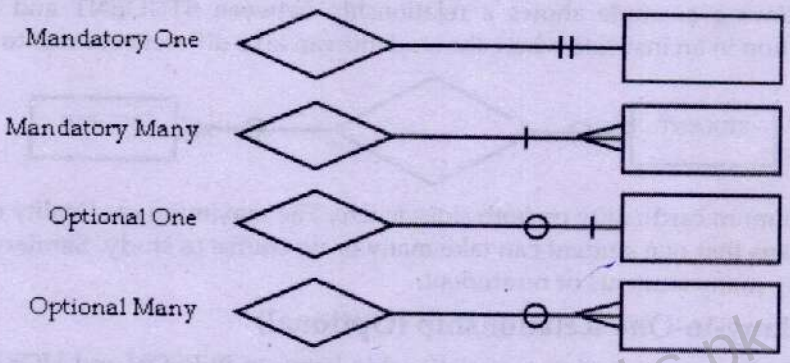

The above relationship means that one or many analysts with one or many programmers work on one or many projects.

## 3.4.4 Cardinality Constraints

The maximum number of relationships is called **cardinality**. The **cardinality constraint** specifies the number of instances of one entity that can be associated with each instance of the other entity. There are three symbols used to show degree. A circle means **zero**, a line means **one** and crow's foot symbol means **many**.

A circle ○ indicates that relationship is **optional**. It means that the minimum number of relationships between each instance of the first entity and instances of the related entity is zero. A stroke | indicates that relationship is **mandatory**. It means that minimum number of relationships between each instance of first entity and instances of the related entity is one.

The second symbol indicates cardinality. A stroke | indicates that maximum number of relationships is one. A crows-foot ⋸ indicates that many relationships between instances of the related entities may exist. These symbols are as follows:




### 3.4.4.1 Minimum Cardinality

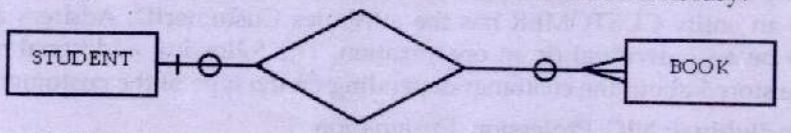
The minimum number of instances of one entity that may be associated with each instance of another entity is known as **minimum cardinality**. If the minimum cardinality is zero, participation is optional. If the minimum cardinality is 1, participation is **mandatory**.

### 3.4.4.2 Maximum Cardinality

The maximum number of instances of one entity that may be associated with each instance of another entity is known as **maximum cardinality**.

### Example: One-to-Many Relationship (Optional)

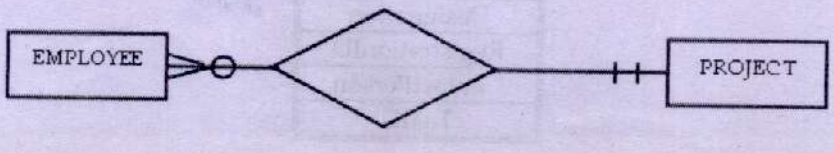
The following example shows a relationship between STUDENT and BOOK. It depitcs a situation in a library where the students can borrow books from the library.




The shape ⊣○ with STUDENT shows that minimum cardinality of STUDENT is zero and maximum is 1. The shape ○⋸ with BOOK shows that there can be zero or many instances of books associated with one student. The minimum cardinality of BOOK is zero and maximum is many. It means that one student can borrow zero or many books and one book can be borrowed by one or no student.

### Example: Many-to-One Relationship (Mandatory)

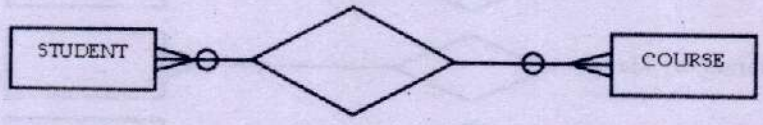
The following example shows a relationship between EMPLOYEE and PROJECT, It depitcs a situation in an organization where employees work on different projects.



The shape ⟩—o with EMPLOYEE shows that minimum cardinality is *zero* and maximum is *many*. The shape ⊢⊣ with PROJECT shows that the minimum and maximum cardinality is 1. There must be exactly one instance of project in the relationship. It means that zero or many employees may be working on one project. But one employee must work on exactly one project.

### Example: Many-to-Many Relationship (Optional)

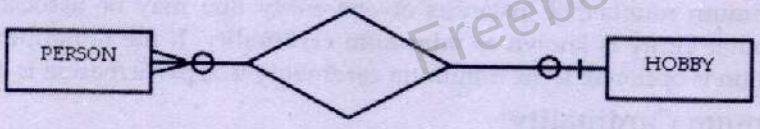
The following example shows a relationship between STUDENT and COURSE. It depitcs a situation in an institute where the students can take different courses to study.




The minimum cardinality on both sides is *zero*. The maximum cardinality on both sides is *many*. It means that one student can take many or no course to study. Similary, one course can be taken by many students or no student.

### Example: Many-to-One Relationship (Optional)

The following example shows a relationship between PERSON and HOBBY. It depitcs a situation in daily life where a person may have no or one hobby.




The minimum cardinality of PERSON is *zero* and maximum is *many*. The minimum cardinality of HOBBY is zero and maximum is *one*. It means that one person can have one or no hobby. Similary, one hobby can be associated with many person or no person.

## 3.5 Subtype & Supertype Entities

An entity that contains some optional attributes or subtypes is called a **subtype entity**. For example, an entity CUSTOMER has the attributes CustomerID, Address and Phone. A customer can be an individual or an organization. The following additional information is required to be stored about the customer depending on the type of the customer:

- **For Individual:** NIC, Profession, Designation
- **For Organization:** RegistrationID, ContactPerson, TaxID

One way to manage this situation is to allocate all above attributes to the entity CUSTOMER. The CUSTOMER contains the following attributes:

| | | | |
|---|---|---|---|
| CustomerID | OP | 0 | M |
| Address | OP | 0 | 1 |
| Phone | Mand | 1 | M |
| NIC | Mand | 1 | 1 |
| Profession | | | |
| Designation | | | |
| RegistrationID | | | |
| ContactPerson | | | |
| TaxID | | | |

If the customer is an individual, RegistrationID, ContactPerson and TaxID are not used. If the customer is an organization, NIC, Profession and Designation are not used.

The second way to manage this situation is to use subtypes as follows:

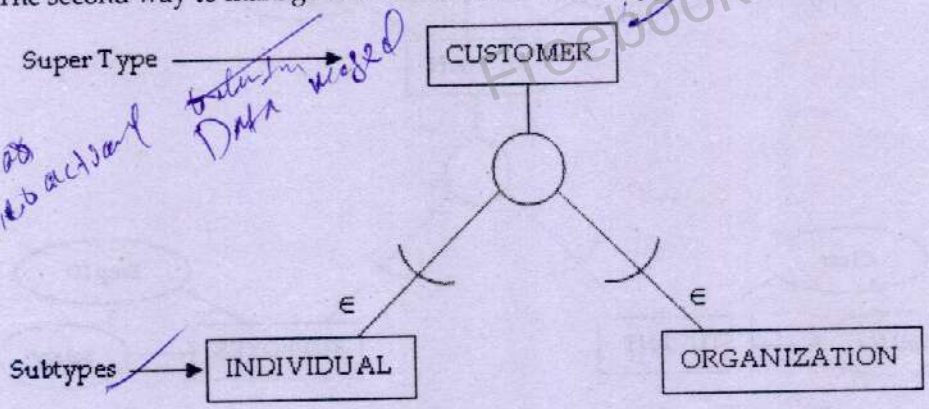



**Figure:** Supertype and Subtype entity

The curve on the line indicates that it is a subtype.

| CUSTOMER contains | INDVIDUAL contains | ORGANIZATION contains |
|---|---|---|
| CustomerID | NIC | RegistrationID |
| Address | Profession | ContactPerson |
| Phone | Designation | TaxID |

In above figure, Individual and Organization are **subtypes** of Customer. It means that Customer is **supertype** of both Individual and Organization. The symbol ε is used to indicate a subtype. The curved line with 1 indicates that Customer must belong to only one subtype.

This structure of super and subtypes are also called **generalization hierarchies** because Customer is a generalization of both Individual and Organization. It is also known as **IS-A relationship** because Individual **is a** Customer and Organization also **is a** Customer.

## 3.5.1 Generalization

Generalization is a process of identifying more general entity types. For example, HUMAN is a more general entity type than STUDENT or LAWYER. The entity type HUMAN contains attributes that are more general than other two entity types. It may contain the attributes like NAME, ADDRESS and AGE etc. The entity type STUDENT contains the attributes which are specific to students like ROLL NO, MARKS and GRADE etc.

## 3.5.2 Specialization

Specialization is a process of identifying more specific entity types. For example, STUDENT and LAWYER are more specific entity types than HUMAN.

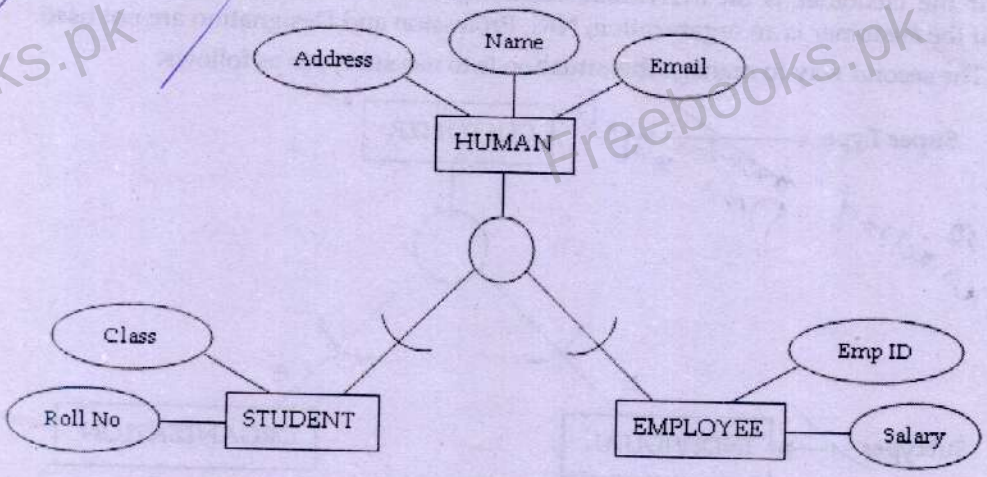



**Figure:** Generalization (Supertype) and Specialization (Subtypes)

In the above figure, HUMAN is supertype and STUDENT and EMPLOYEE are its two subtypes. The subtypes also contain attributes of supertype along with their own additional attributes. The subtypes are identified from one supertype with the process of **specialization**.

## E-R Project 1

Draw an ER diagram for the following situations. Identify cardinality, existence and optionality for subtypes of each relationship.

1. A company has a number of employees. Each employee may be assigned to one or more projects or may not be assigned to a project. A project must have at least one employee assigned and may have several employees assigned.


2. A hospital patient has a patient history. Each patient has one or more history records. Each patient history record belongs to exactly one patient.


3. An account can be charged against many projects though it may not be charged against any. A project must have at least one accounts charged against it. It may have many accounts charged against it.

4. An employee must manage exactly one department. A department may or may not have one employee to manage it.


## E-R Project 2

Identify two entity types and one relationship for each of the following pairs of rules. State the cardinality and existence of the relationship in each case. Draw the ER diagram.

**1. A department employs many persons. A person is employed by one department at most.**

**Solution:** The following diagram shows that a department employees one or many persons. A person may be employed by one department or he may not be employed at all.


**2. A manager manages one department at most. A department is managed by one manager at most.**

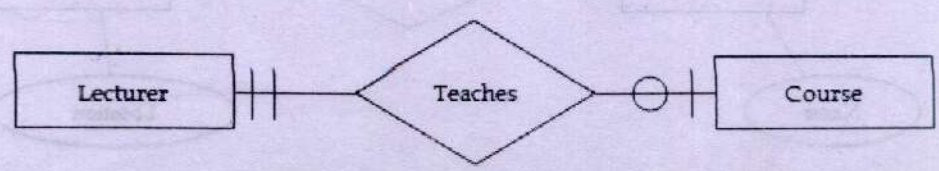
**Solution:** The following diagram shows that a manager may manage one department or he manages no department. A department is managed by one manager or it it is not managed by any manager.


**3. A team consists of many players. A player plays for only one team.**

**Solution:** The following diagram shows that a team employs at least one player or many players. A player plays for exactly one team.


**4. A lecturer teaches one course at most. A course is taught by exactly one lecturer.**

**Solution:** The following diagram shows that a lecturer teaches one course or he does teach any course. A course is taught by exactly one lecturer.



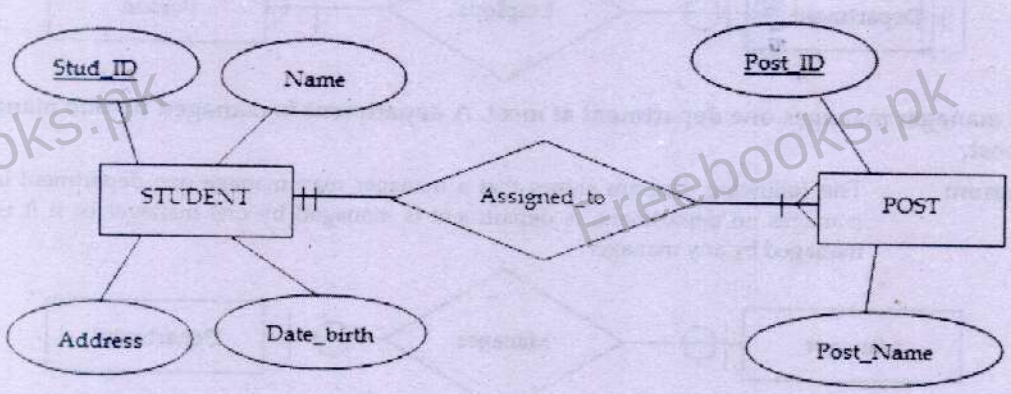
**5. A purchase order may be for many products. A product may appear on many purchase orders.**

**Solution:** The following diagram shows that a purchase order contains one or many products. A product may appear in many orders and it may not appear in any order at all.


## E-R Project 3

In a school, a student may be assigned to one or more posts like perfect, monitor or chairman. A post must be assigned to exactly one student. A student is identified with student ID, name, address and date of birth. A post is identified with post ID and name. Draw ER diagram to represent relationship between STUDENT and POST with cardinality.

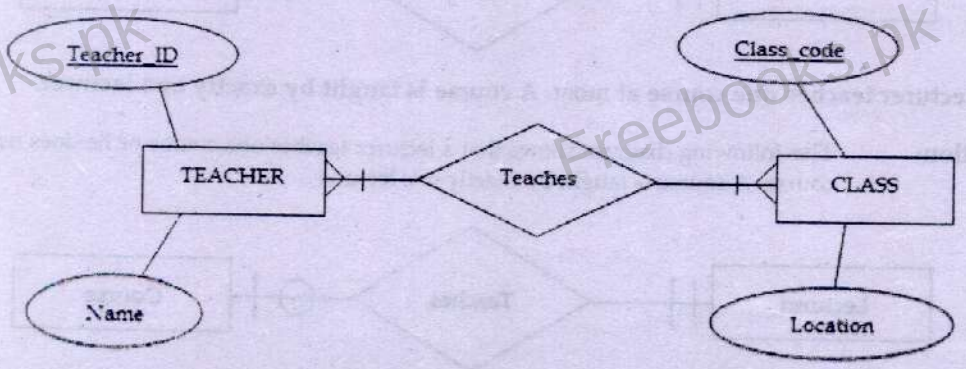
## Solution




## E-R Project 4

In a school, a teacher teaches one or more classes. each class is taught by one or more teachers. A teacher is identified with teacher's ID and name. A class is identified with class code and location. Draw an ER diagram to represent the relationship between TEACHER and CLASS indicating cardinality.

## Solution

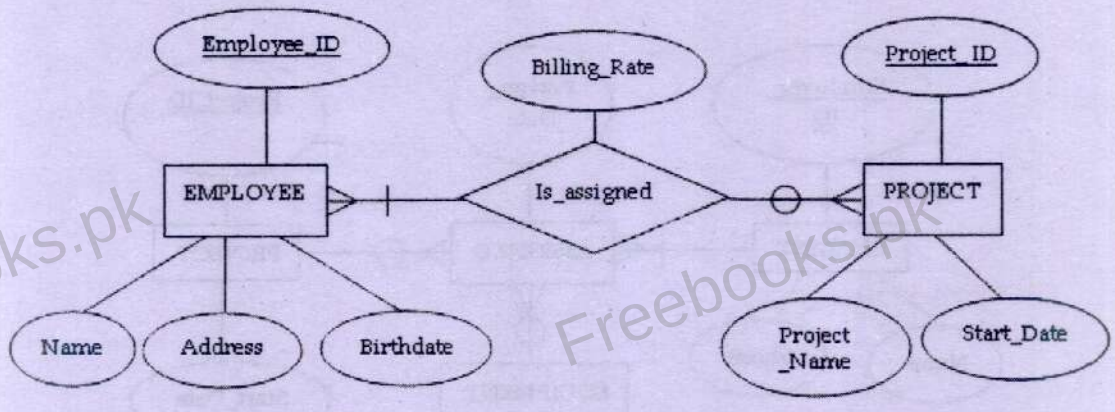


## E-R Project 5

Draw the E-R model for the following scenario:

A company has a number of employees. The attributes of EMPLOYEE include Employee_ID (identifier), Name, Address and Birthdate. The company also has several projects. The attributes of PROJECT are Project_ID (identifier), Project_Name and Start_Date. Each employee may be assigned to one or more projects or may not be assigned to a project. A project must have at least one employee assigned and may have any number of employees assigned. An employee's billing rate may vary by project and the company wishes to record the applicable billing rate (Billing_Rate) for each employee when assigned to a particular project.

## Solution

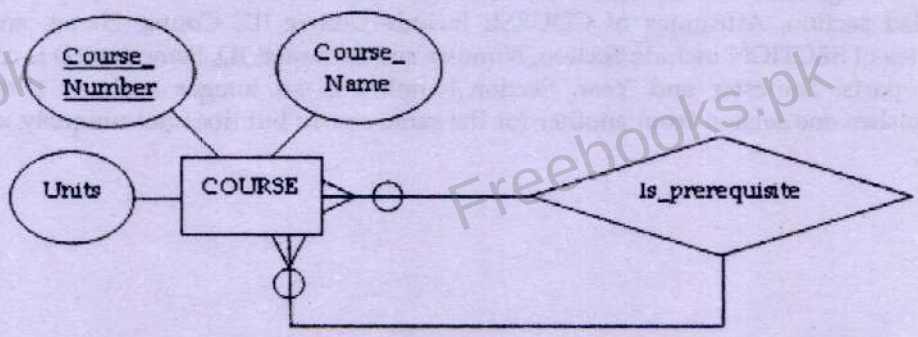



## E-R Project 6

Draw E-R model for the following scenario:

A university has a large number of courses in catalog. Attributes of COURSE include Course_Number (identifier), Course_Name and Units. Each course may have one or more courses as prerequisites or may have no prerequisites. Similarly, a particular course may be a prerequisite for any number of courses or may not be prerequisite for any other course.

## Solution

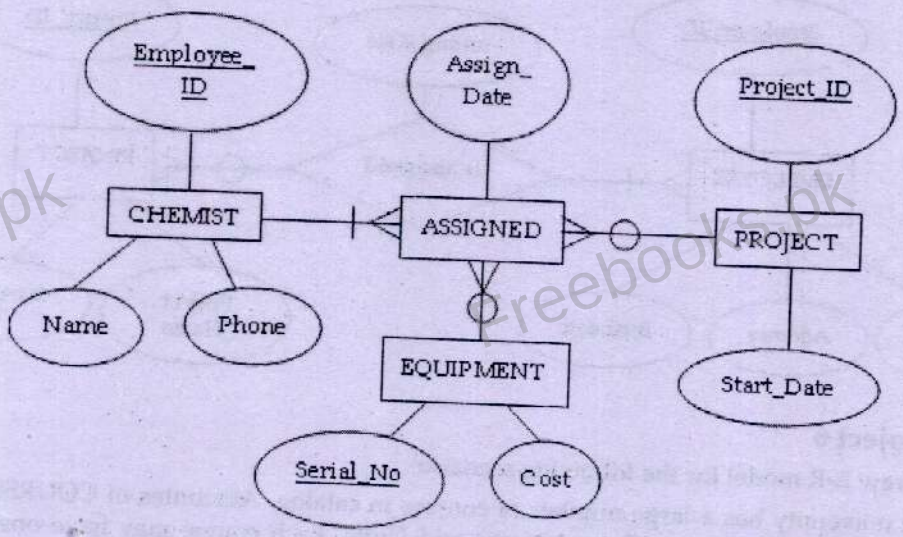


## E-R Project 7

Draw E-R model for the following scenario:

A laboratory has several chemists who work on one or more projects. Chemists also may use certain kinds of equipment on each project. Attributes of CHEMIST include Employee_ID (identifier), Name and Phone_No. Attributes of PROJECT include Project_ID (identifier) and Start_Date. Attributes of EQUIPMENT include Serial_No and Cost. The organization wishes to record Assign_Date i.e. the date when an equipment item was assigned to a particular chemist working on a specific project. A chemist must be assigned to at least one project and one equipment item. An equipment item need not be assigned and a given project need not be assigned either a chemist or an equipment item.

## Solution

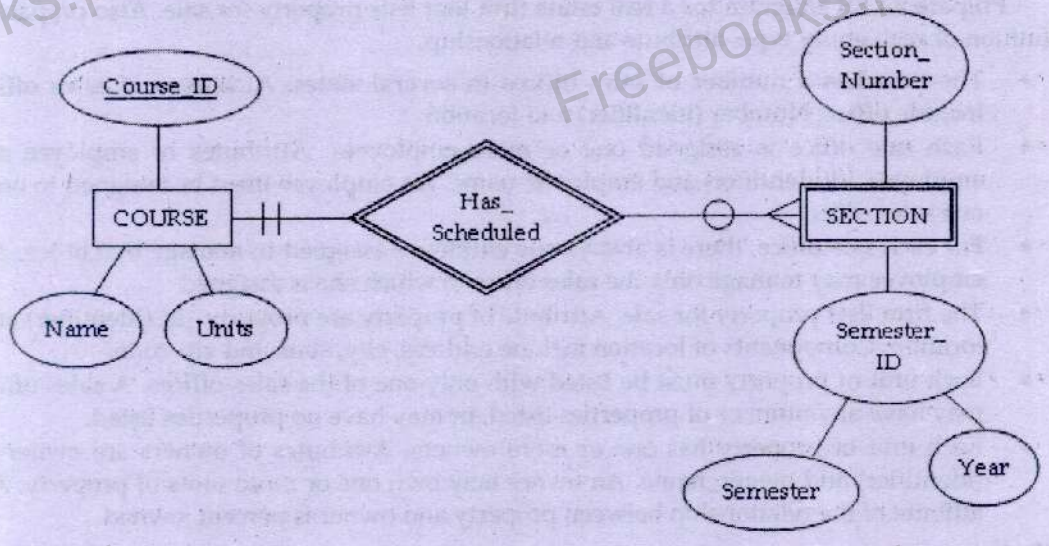



## E-R Project 8

Draw E-R model for the following scenario:

A college course may have one or more scheduled sections or may not have a scheduled section. Attributes of COURSE include Course_ID, Course_Name, and Units. Attributes of SECTION include Section_Number and Semester_ID. Semester_ID is composed of two parts: Semester and Year. Section_Number is an integer such as 1 or 2 that distinguishes one section from another for the same course but does not uniquely identify a section.

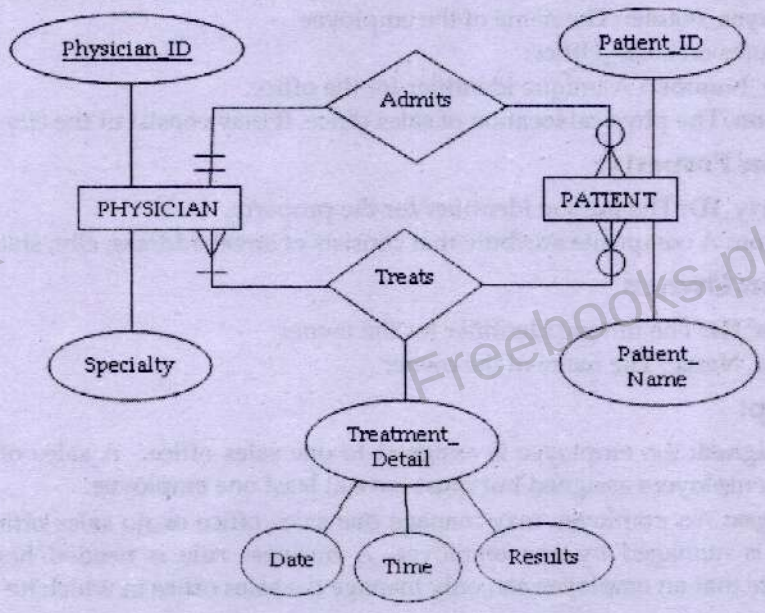
## Solution




## E-R Project 9

A hospital has a large number of registered physicians. Attributes of PHYSICIAN include Physician_ID (identifier) and Specialty. Patients are admitted to the hospital by physicians. Attributes of PATIENT include Patient_ID (identifier) and Patient_Name. Any admitted must have exactly one admitting physician. A physician may admit any number of patients. Once admitted, a given patient must be treated by at least one physician. A particular physician may treat any number of patients or may not treat any patients. Whenver a patient is treated by a physician, the hospital records the details of the treatment (Treatment_Detail). Components of Treatment_Detail include Date, Time, and Results.

## Solution



## E-R Project 10

Prepare an ER Diagram for a real estate firm that lists property for sale. Also prepare a definition of each entity type, attribute and relationship.

- The firm has a number of sales offices in several states. Attributes of sales office include office_Number (identifier) and location
- Each sale office is assigned one or more employees. Attributes of employee are employee_id(identifier) and employee_name. An employee must be assigned to only one sales office.
- For each sale office, there is always one employee assigned to manage that office. An employee may manage only the sales office to which she is assigned
- The firm lists property for sale. Attribute of property are property _id (Identifier) and location. Components of location include address, city, state and zip-code.
- Each unit of property must be listed with only one of the sales offices. A sales office may have any number of properties listed, or may have no properties listed.
- Each unit of property has one or more owners. Attributes of owners are owner-id (identifier) and owner_name. An owner may own one or more units of property. An attibute of the relationship between property and owner is percent_owned.

## Solution

### Entities:

- **Employee:** An employee of the firm works for one sales office and may manage one sales office. It is not indicated that employee can manage the office that he works in. This would require a business rule.
- **Sales_Office:** The office where real estate is sold.
- **Property:** Buildings for sale, such as houses, condos and apartment buildings.
- **Owner:** The individual who owns one or more properties.

### Attributes on Employee:

- **Employee_ID:** A unique identifier for an employee. This attribute must be unique.
- **Employee_Name:** The name of the employee
- Attributes on Sales_Office:
- **Office_Number:** A unique identifier for the office.
- **Location:** The physical location of sales office. It may consist of the city and state.

### Attributes on Property:

- **Property_ID:** The unique identifier for the property.
- **Location:** A composite attribute that consists of street address, city, state and Zipcode

### Attributes on Owner:

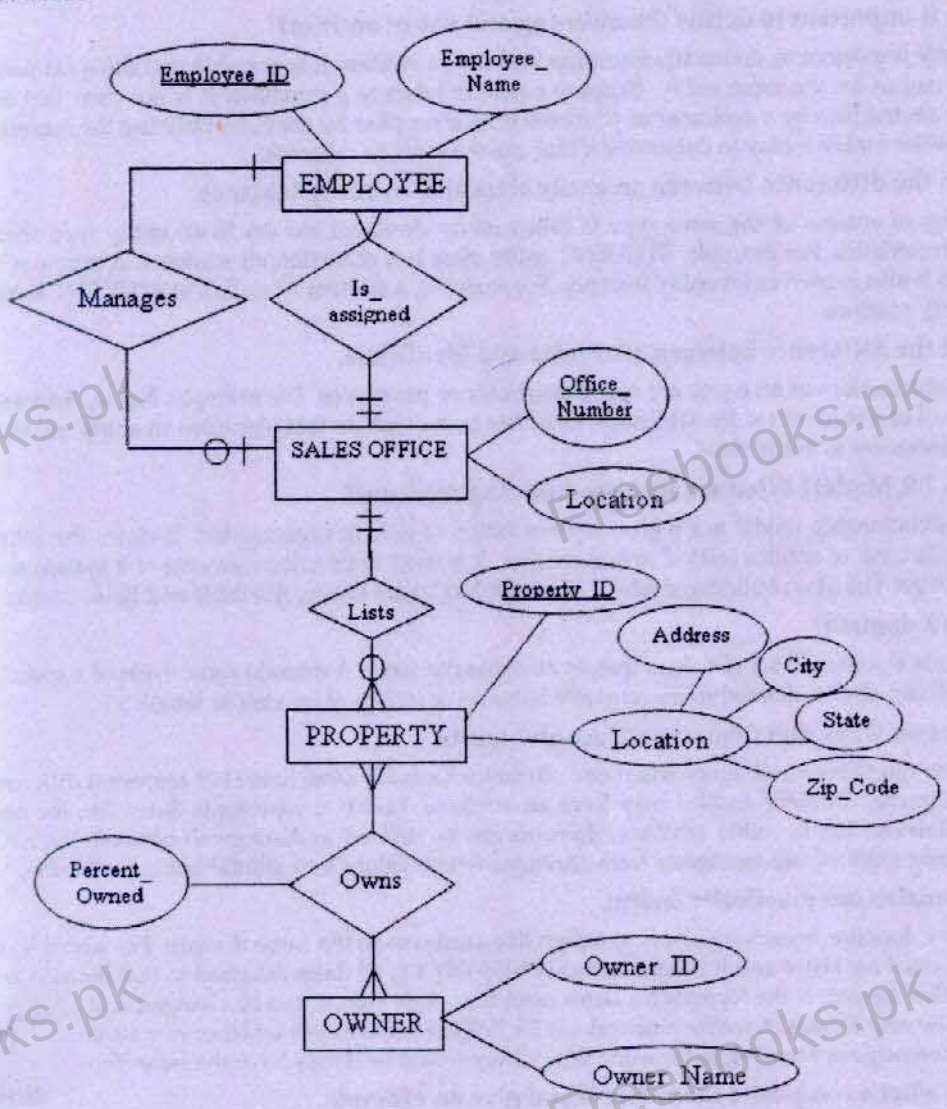
- **Owner_ID:** The unique identifier for the owner.
- **Owner_Name:** The name of the owner

### Relationship:

- **Is_assigned:** An employee is assigned to one sales office. A sales office may have many employees assigned but must have at least one employee.
- **Manages:** An employee may manage one sales office or no sales office. Each sales office is managed by one employee. A business rule is needed here in order to indicate that an employee can only manage the sales office in which he works.

- **Lists:** Each property is listed by only one sales office. Each sales office can list one, none or many properties.
- **Owns:** Each property has one or more owners. Each owner can own one or more properties. Percent_owned is an attribute on Owns and tracks the percent of the property that the owner owns.

**Solution**



## Short Questions

**Q.1. Define entity and give an example.**

An entity is a person, place, thing or event for which data is collected and maintained. For example, a library system may contain data about different entities like BOOK and MEMBER. A college system may include entities like STUDENT, TEACHER, and CLASS etc.

**Q.2. Why is it important to define the meaning and use of entities?**

It is very important to define the meanings and use of entities. It is possible that different users use different names for the same entity. Suppose a person refers to a purchase. It is not clear that the purchase is from the firm by a customer or purchase from a supplier by the firm. Defining the meaning and use of entities makes it easy to determine if they are the same or different.

**Q.3. Explain the difference between an entity class and an entity instance.**

A group of entities of the same type is called entity class. All entities in an entity type share common characteristics. For example, STUDENT entity class is a collection all students. A member of an entity class is also known as an entity instance. For example, a student Abdullah of STUDENT entity type is an entity instance.

**Q.4. Explain the difference between attributes and identifiers.**

The characteristics of an entity are called **attributes** or **properties**. For example, Name, Address, Class and Email of a student are his attributes. Identifier is an attribute that identifies an entity instance among other instances in entity class.

**Q.5. What is ER Model? What are its main building modules?**

Entity-Relationship model is a logical representation of data in organization. It views the entire system as a collection of entities related to one another. It is used to describe elements of a system and their relationships. The main building modules of the ER Model are Entity, Attribute and Relationship.

**Q.6. What is a domain?**

A domain is a named set of values that an attribute can have. A domain can consist of a specific list of values. It can also be defined more generally like a set of strings of maximum length 50.

**Q.7. Explain two ways that domains reduce ambiguity.**

Domains can reduce ambiguity when two attributes have the same name but represent different things. For example, different entities may have an attribute Tax. If it represents Sales Tax for one attribute and Income Tax for other attribute, domains can be defined to distinguish between the two. The named domains eliminate ambiguity from attributes whose values look similar but are not same.

**Q.8. How domains are practically useful.**

One way domains become useful is to assign like attributes to the same domain. For example, if a domain is created for Dates and it used a format of MM-DD-YY, all dates assigned to that domain are forced to use that format. If the formats for dates need four-digit year, it can be changed and all dates will now use the new format. Another practical use for domains is to assess whether two attributes that are named differently are referring to the same thing. They would be if they have the same domain.

**Q.9. Explain what a composite identifier is and give an example.**

An identifier that consists of composite attribute is called **composite identifier**. For example, OrderID identifier may consist of OrderNo and Date.

**Q.10. What do you know about simple and Composite Attribute. Give example of each.**

An attribute that cannot be subdivided into smaller component is known as **simple attribute**. For example, a person can have only one gender and one date of birth. An attribute that can be subdivided into smaller components is called **composite attribute**. For example, Address is an example composite attribute. It can be subdivided into Street, City, Country.

**Q.11. What is difference between single-valued and multi-valued attribute. Give example.**

An attribute that may contain single value is called **single-valued attribute**. For example, Age of a person is single-valued attribute. An attribute that may contain two or more values is called **multi-valued attribute**. For example, a person can have two or more college degrees.

**Q.12. Give an example of an entity that has no obvious identifier.**

A very simple example is a house or address. Initially you may think that the phone number of a house would be an identifier but some houses may not have phones and all instances of the entity must have an identifier. Next you may think house number and street name but what about rural areas. These areas typically have Route Numbers but the houses do not have house numbers. Then you could use Route Numbers and P.O. Box but houses in the city do not have P.O. Boxes.

**Q.13. Define relationship and give an example.**

A **relationship** is a logical connection between different entities. The entities that participate in a relationship are called **participants**. The relationship may be between different entities or between an entity and itself. A relationship is established on the basis of interaction among entities. For example, a relationship exists between a STUDENT and TEACHER because the teacher *teaches* the students.

**Q.14. Define degree of relationship. Give example of a relationship greater than degree 2.**

The number of entities in a relationship is called **degree** of relationship. Assume entity classes FACULTY, STUDENT and MAJOR. Assume FACULTY is assigned to advise STUDENT for a given MAJOR.

**Q.15. Explain the difference between a relationship class and a relationship instance.**

A relationship class is an association among entity classes. A relationship instance is an association among entity instances.

**Q.16. Define the terms maximum cardinality and minimum cardinality.**

The minimum number of instances of one entity that may be associated with each instance of another entity is known as **minimum cardinality**. The maximum number of instances of one entity that may be associated with each instance of another entity is known as **maximum cardinality**.

**Q.17. Define the term weak entity and give an example.**

An entity that can exist only if another entity exists is known as weak entity. It means that weak entities depend upon the existence of another entity. Suppose we want to store the data of a student after assigning a class to him. It means that the data cannot be stored if CLASS entity does not exist. In order to store the record of the student, we first need to create an entity that represents a class. Here, STUDENT is a weak entity because it depends upon CLASS entity.

**Q.18. Differentiate between single-valued attributes and simple attributes with example.**

A single-valued attribute is an attribute that can have only one value. For example, a person has only one first name and only one social security number. A simple attribute is an attribute that cannot be decomposed. For example, a person's gender can be either M or F. It cannot be decomposed. Single-valued attributes are not necessarily simple. For example, an inventory code HWPR145 may refer to a classification scheme. HW indicates HardWare, PR indicates Printer, and 145 indicates an inventory number. It can be decomposed into its component parts even if it is single-valued.

**Q.19. What cardinalities indicate functional, optional and mandatory relationships?**

A maximum cardinality of one indicates a functional relationship. A minimum cardinality of zero indicates an optional relationship. A minimum cardinality of one or more indicates a mandatory relationship.

**Q.20. What is the difference between an existence-dependent and a weak entity type?**

A weak entity is a specialized kind of existence-dependent entity. A weak entity has mandatory relationship like an existent-dependent entity. In addition, weak entities borrow part or their entire primary key.

**Q.21. When should an ERD contain weak entities?**

An ERD contains weak entity types when entities are closely associated with other entities. Identification dependency often occurs when entities are physically located inside another entity such as rooms inside a building.

**Q.22. Describe subtype entities and give an example.**

A subtype entity is an entity that represents a special case of another entity called its supertype. For example, the entity STUDENT can have subtypes of GRADUATE_STUDENT and UNDERGRADUATE_STUDENT. The entity BUILDING can have subtypes of CLASSROOM, OFFICE, or RECREATIONAL etc.

**Q.23. Differentiate between a HAS-A and an IS-A relationship with example.**

The relationship between a supertype and its subtypes is also called an IS-A relationship. Entities with an IS-A relationship should have the same identifier as they represent different aspects of the same thing. Entities with an HAS-A relationships represent aspects of different things. They have different identifiers. These relationships do not involve subtypes.

BUILDING with an identifier of Building Code and subtypes of CLASSROOM, OFFICE and RECREATIONAL have an IS-A relationship. All types of building are identified by building code. All subtypes IS-A BUILDING. A relationship between ADVISOR and STUDENT is a HAS-A relationship because a STUDENT HAS-A ADVISOR, ADVSIOR is not a type of STUDENT.

**Q.24.Constract the following terms:**

a. Entity type and relationship type b. Degree and cardinality

a. An entity type is a collection of entities that share common properties or characteristics. A relationship type is a meaningful association between entity types.

b. The degree (of a relationship) is the number of entity types that participate in that relationship. Cardinality is a constraint on the number of instances of one entity that can (or must) be associated with each instance of another entity.

**Q.25. Name the symbols used in E-R model for (a) entity, (b) relationship, (c) weak entity and its relationship, (d) recursive relationship and (e) subtype entity.**

(a) Rectangle (b) Diamond (c) Double-lined rectangle and double-lined diamond
(d) Line with diamond back to entity (e) Entity with existence symbol

**Q.26. List four types of Cardinality constraint and draw an example of each.**

a. Optional one:


b. Mandatory one:

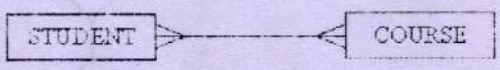

c. Optional many


d. Mandatory many

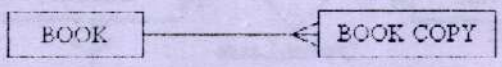
**Q.27. For each of the following, indicate whether there is a one-to-many or a many-to-many relationship. Also draw a diagram of the following relationship.**

a. STUDENT and COURSE (student register for course)
b. BOOK and BOOK COPY (book have copies)
c. COURSE and SECTION(courses have section)
d. SECTION and ROOM( sections are scheduled in rooms)
e. INSTRUCTOR and COURSE

a. Many-to-many




b. One-to-many

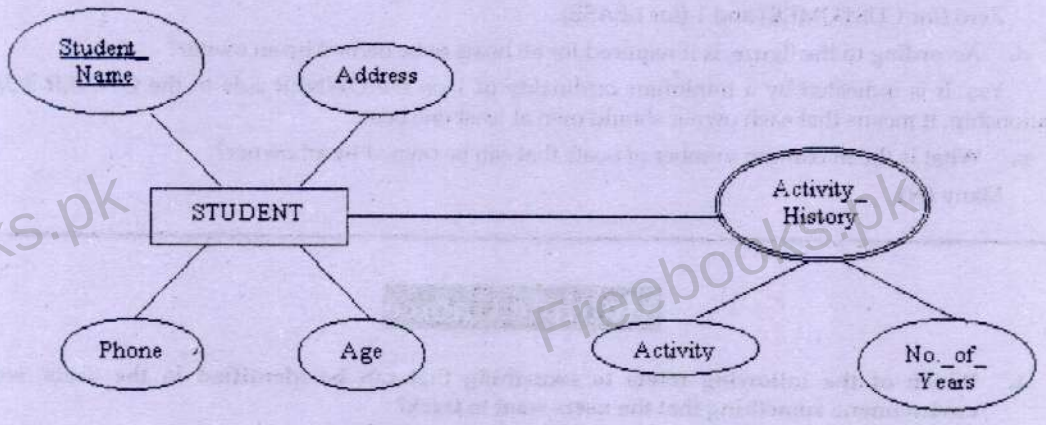



c. One-to-many


d. One-to-many


e. Many-to-many

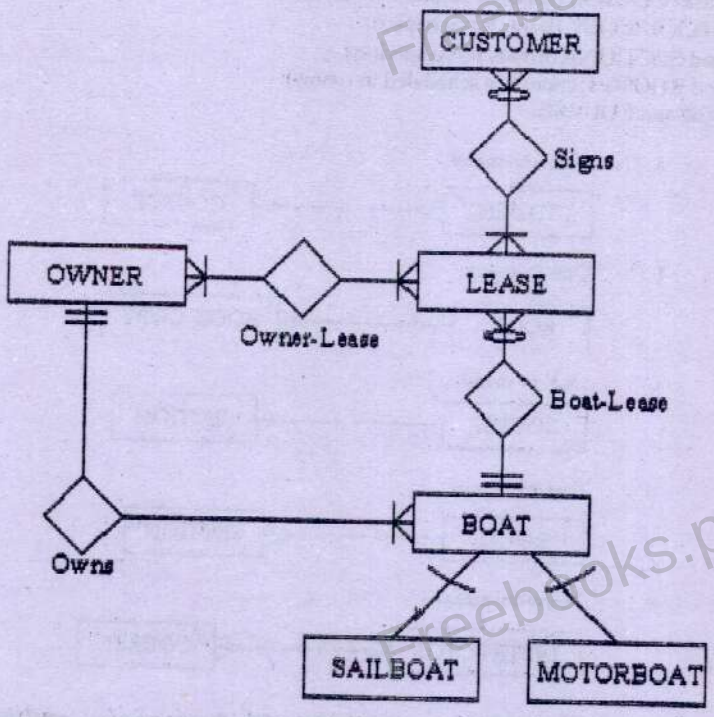

**Q.28. The entity type STUDENT has attributes Student_name, address, phone, age, activity and no_of_years. Activity represents campus-based student activity while no_of_years represents the number of years student has engaged in this activity. A given student may engage in more than one activity. Draw an E-R Diagram for this situation.**




**Notes:**

1. Assume that Student_Name is the identifier.
2. Age cannot be calculated without date-of-birth

**Q.27. Examine the following entity-relationship diagram and answer the questions that follow. The arcs imply the disjoint sub type/super type relation.**




a. List one example of a one-to-many relationship.

A one-to-many relationship exists between OWNER and BOAT.

b. What is the degree of relationship between CUSTOMER and LEASE?

Binary, many to many relationship

c. What is the minimum cardinality of the relationship between CUSTOMER and LEASE?

Zero (for CUSTOMER) and 1 (for LEASE).

d. According to the figure, is it required for all boats to be owned by an owner?

Yes. It is indicated by a minimum cardinality of 1 on the OWNER side in the OWNER-BOAT relationship. It means that each owner should own at least one boat.

e. What is the maximum number of boats that can be owned by an owner?

Many (N).

## Multiple Choice

1. **Which of the following refers to something that can be identified in the users' work environment, something that the users want to track?**
   a. Entity b. Attribute c. Identifier d. Relationship
2. **Properties that describe the entity's characteristics are called:**
   a. Entity b. Attribute c. Identifier d. Relationship
3. **Attributes that name, or identify, entity instances are called:**
   a. Entity b. Attribute c. Identifier d. Relationship

4. **Customers, cars, and parts are examples of:**
   a. Entities b. Attributes c. Cardinals d. Relationships
5. **Which of the following represent the entities?**
   a. Teacher b. Student c. Aero plane d. All
6. **An identifier may be:**
   a. composite b. Unique c. non-unique d. All
7. **Entities can be associated with one another in which of the following:**
   a. Entity b. Attribute c. Identifier d. Relationship
8. **The relationship can be:**
   a. one to one b. one to many c. Many to many d. All
9. **In which of the following is a single-entity instance of one type related to a single-entity instance of another type?**
   a. One-to-one relationship b. One-to-many relationship
   c. Many-to-many relationship d. Composite relationship
10. **In which of the following is a single-entity instance of one type related to many entity instances of another type?**
    a. One-to-one relationship b. One-to-many relationship
    c. Many-to-many relationship d. Composite relationship
11. **In which of the following can many entity instances of one type be related to many entity instances of another type?**
    a. One-to-one relationship b. One-to-many relationship
    c. Many-to-many relationship d. Composite relationship
12. **Which of the following is an example of one-to-one relationship?**
    a. Student-RegNo b. Person-automobile
    c. Mother-daughter d. Person-phone number
13. **Which of the following is a one-to-Many relationship?**
    a. Student-RegNo b. Person-automobile
    c. Mother-daughter d. Both b and c
14. **A relationship between countries and capitals is an example of _____ relationship:**
    a. One-to-One b. One-to-Many c. Many-to-Many d. Many-to-One
15. **Which of the following is related to Modality?**
    a. Optional b. Mandatory c. Unidirectional d. Both a and b
16. **An entity related to itself in an ERD model refers to:**
    a. Recursive relationship b. One-to-many relationship
    c. Many-to-many relationship d. One-to-one relationship
17. **A data model is:**
    a. A logical representation of the structure of the database
    b. Shown as an entity-relationship diagram
    c. Transformed into tables and relationships
    d. All
18. **Which of the following indicates the maximum number of entities that can be involved in a relationship?**
    a. Minimum cardinality b. Maximum cardinality c. E-R diagrams d. Greater entity count
19. **Which of the following refers to an entity that logically depends on another entity?**
    a. Weak entity b. Strong entity c. ID-dependent entity d. Dependent
20. **In an E-R diagram, a rectangle represents a(n):**
    a. Entity class b. Weak entity c. Relationship d. Attribute
21. **In an E-R diagram, a rectangle with rounded corners represents a(n):**
    a. Entity class b. Weak entity c. Relationship d. Attribute

22. **In an E-R diagram, a ellipse represents a(n):**
a. Entity class b. Weak entity c. Relationship d. Attribute

23. **In an E-R diagram a relationship is represented by a(n):**
a. Rectangle b. Ellipse c. Rectangle with rounded corners d. Diamond

24. **An attribute which consists of a group of attributes is called:**
a. Multi-valued attributes b. Composite attributes c. Composite identifiers d. Identifiers

25. **Identifiers that consist of two or more attributes are called:**
a. Multi-valued attributes b. Composite attributes c. Composite identifiers d. Identifiers

26. **Which refers to connecting entities of different types when identifiers are different?**
a. HAS-A relationships b. IS-A relationships c. Binary relationships d. None

27. **Which of the following refers to relationships that are subtypes, when the identifiers of the entities are the same?**
a. HAS-A relationships b. IS-A relationships c. Binary relationships d. None

28. **Relationships among entities of a single class are called:**
a. HAS-A relationships b. IS-A relationships c. Recursive Relationship d. None

29. **Which is NOT included in the definition of an entity?**
a. Person b. Object c. Concept d. Action

30. **Which is NOT an example of a strong entity type?**
a. Student b. Course c. Department d. Student_id

31. **If EMPLOYEE is the entity type, then SMITH, JOHN is the entity ____:**
a. Field b. Characteristics c. Identifier d. Instance

32. **A meaningful association between entity types is a(n)**
a. Entity identifier b. Relationship type
c. Relationship instance d. Associative entity

33. **An entity type whose existence depends on another entity type is called _____ entity**
a. Strong b. Weak c. Dependent d. Variant

34. **An entity that associates the instances of one or more entity types and contains attributes specific to the relationships is called a (n)**
a. Associative entity b. Connecting entity c. Intersectional entity d. All

35. **A persons name, birthday, and social security number are all examples of**
a. Attributes b. Entities c. Relationships d. Descriptors

36. **The most common type of relationship encountered in data modeling is ____ relationship:**
a. Unary b. Binary c. Ternary d. Associative

37. **Which is NOT a basic construct of an E-R model?**
a. Relationships b. Entity types c. Identifiers d. Attributes

38. **An example of a multi-valued attribute might be**
a. Student_Address b.College_Degree c. Student_GPA. d. ID_Number.

39. **_____ specifies the number of instances of one entity that can (or must) be associated with each instance of another**
a. Multivalued attribute b. Cardinality constraint c. Entity instance d. associative entity

40. **An entity class:**
a. Contains one instance of a particular entity b. Contains structure or format of entity
c. Represents something that the users want to track d. b and c

41. **All instances of a given entity:**
a. Have the same values for the attributes b. Belong to the same entity class
c. Have the same attributes d. b and c

42. **An identifier of an entity instance:**
a. May be unique b. Must be unique
c. May consist of more than one attribute d. a and c

**43. Which type of identifier is used to identify a set of instances of a given entity?**
a. Unique b. Non-unique c. Instance d. None

**44. Depending on the system being modeled, a relationship can have? entities:**
a. One b. Two c. Three d. All

**45. The degree of a relationship refers to the:**
a. Number of entities b. Maximum cardinality
c. Minimum cardinality d. Number of attributes in the identifiers

**46. A weak entity is one which:**
a. Is not in a relationship with any other entities b. Does not have a unique identifier
c. Cannot exist in the database by itself d. Is a subtype

**47. A subtype entity:**
a. Inherits attributes of its supertype
b. Contains optional attributes not contained in supertype
c. Is always mutually exclusive
d. a and b

**48. Cardinality expresses ____ number of entity occurrences associated with one occurrence of the related entity.**
a. Undetermined b. the specific c. Pre-determined d. Programmed

**49. Knowing ____ number of entity occurrences is very helpful at application software level.**
a. Maximum b. Minimum c. Exact d. Maximum and minimum

**50. A ____ attribute need not be physically stored within the database.**
a. composite b. Multivalued c. single-valued d. derived

**51. A ____ relationship exists when three entities are associated.**
a. unary b. Binary c. Ternary d. weak

**52. Attributes may share a:**
a. name b. Domain c. Location d. table

**53. Which of the following might be represented with a single-valued attribute?**
a. Person's phone number(s) b. Car's color
c. Employee's educational background d. Computer's processor speed

**54. If an entity can exist apart from one or more related entities, it is said to be ____-independent.**
a. existence b. Relationship c. Business d. weak

**55. A relationship is an association between ____.**
a. objects b. Entities c. Databases d. fields

## Answers

| | | | | | |
|---|---|---|---|---|---|
| 1. a | 2. b | 3. c | 4. a | 5. d | 6. d |
| 7. d | 8. d | 9. a | 10. b | 11. c | 12. a |
| 13. d | 14. a | 15. d | 16. a | 17. d | 18. b |
| 19. a | 20. a | 21. b | 22. d | 23. d | 24. a |
| 25. b | 26. a | 27. b | 28. c | 29. d | 30. d |
| 31. d | 32. b | 33. b | 34. a | 35. a | 36. b |
| 37. c | 38. b | 39. b | 40. d | 41. d | 42. d |
| 43. b | 44. d | 45. a | 46. c | 47. d | 48. b |
| 49. d | 50. d | 51. c | 52. b | 53. d | 54. a |
| 55. b | | | | | |

## True / False

1. An E-R diagram is an example of a physical schema.
2. The E-R model was introduced in an article by Chen in the 1980s.
3. An E-R diagram consists of entities and resources.
4. A strong entity type does not need an identifier.
5. STUDENT_REGISTRATION_FOR_CLASS is a good entity type name.
6. A cardinality constraint specifies the maximum number of attributes an entity may have.
7. An entity can have only one defined attribute.
8. A multivalued attribute is an attribute that is common to many entities.
9. A relationship is an association between attributes.
10. The degree of a relationship is the number of entity types that participate in a relationship.
11. A ternary relationship is a simultaneous relationship between more than three entity types.
12. A relationship class involves only one entity class.
13. Relationship classes are associations among entity instances.
14. Relationship instances are associations among entity classes.
15. The number of relationship classes in the relationship is the degree of the relationship.
16. Binary relationships are relationships of degree 2.
17. Identifiers that consist of two or more attributes are called composite identifiers.
18. An identifier must be unique.
19. An entity class is a collection of entities and is described by the structure or format of the entities in that class.
20. An instance of a class is the representation of a particular entity; it is described by the values of its attributes.
21. Recursive relationships are relationships among entities of a single class.
22. Constraints showing the maximum number of entities that can occur on one side of the relationship are called the relationship's maximum cardinality.
23. An entity that is not weak is called a strong entity.
24. Inheritance means that the entities in subtypes inherit attributes of the supertype entity class
25. Class attributes are the same as entity attributes.
26. In an E-R diagram, entities are shown as rectangles with rounded corners.
27. HAS-A relationships are subtypes, and the identifiers of the entities are the same.
28. IS-A relationships connect entities of different types, and the identifiers of the entities are different.
29. Tools for building E-R diagrams are included in most CASE products.
30. Subtype entities are defined when an entity can have optional attributes.
31. Subtype entities have the same identifier as their supertype.
32. The entities in subtypes inherit the attributes of the supertype entity class.
33. The maximum number of entities in a relationship is two.
34. The minimum number of entities in a relationship is two.
35. Entities are associated with one another in relationships.
36. Relationships cannot have attributes.
37. Each instance of an entity has at least one attribute used as an identifier.
38. Every identifier of an entity must be unique.
39. An identifier can consist of only one attribute.
40. There is one generally accepted standard E-R model.
41. A particular occurrence of an entity is known as an instance.
42. An entity's characteristics are described by its attributes.

43. All instances of a given entity class do not have to have the same attributes.
44. The ER diagram represents the conceptual database as viewed by the end user.
45. Attributes do not have a domain.
46. Relationship participation is not very important when designing a database.

## Answers

| | | | |
|---|---|---|---|
| 1. F | 2. F | 3. T | 4. F |
| 5. F | 6. F | 7. F | 8. F |
| 9. F | 10. T | 11. F | 12. F |
| 13. F | 14. F | 15. T | 16. T |
| 17. T | 18. F | 19. T | 20. T |
| 21. T | 22. T | 23. T | 24. T |
| 25. F | 26. F | 27. F | 28. F |
| 29. T | 30. T | 31. T | 32. T |
| 33. F | 34. F | 35. T | 36. F |
| 37. T | 38. F | 39. F | 40. F |
| 41. T | 42. T | 43. F | 44. T |
| 45. F | 46. F | | |

CHAPTER 4

# Semantic Object Model

## Chapter Overview

# 4.1 Semantic Object Model

Semantic object model is a logical representation of data in an organization. It views the entire system as a collection of objects related to one another. Semantic object model takes semantic object as the basic element. It is based on the concepts presented by Codd. It was first presented in 1988. Semantic object model can represent the perceptions of the user more closely than E-R model.

# 4.2 Semantic Objects

A semantic object is a thing that can be identified in the user's working environment. Semantic means "meaning". Semantic objects are used to represent or model the meaning of user's data. For example, different objects in a university are students, teachers, and departments etc.

Semantic objects are grouped into classes. Each object class has a unique name. For example, different classes in a university are STUDENT, TEACHER and DEPARTMENT etc.

A particular semantic object is called an **instance** of the class. For example, "Usman" is an instance of STUDENT class, "Abdullah" is an instance of TEACHER class and "Computer Science" is an instance of DEPARTMENT class.

Some objects exist physically but some do not. For example, a STUDENT is a physical object but ORDER is not a physical object. It is simply a written document. When objects in a working environment are defined, all physical and non physical objects must be identified.

## 4.2.1 Attributes

An object has many attributes. Each attribute represents a characteristic of the object. For example, an object of STUDENT class can have attributes like Name, Address, Email, and Phone etc. These attributes are used to describe the object. When we define objects for a database system, we use only those attributes, which are important for the system. The remaining attributes are ignored. For example, weight is an attribute of a student but it is normally not used in a college database because it is not important.

Different types of attributes are as follows:

### 4.2.1.1 Simple Attributes

Simple attributes are the attributes that contain a single value. Roll No, Marks and Salary are examples of simple attributes.

### 4.2.1.2 Group Attributes

Group attributes are composites of other attributes. For example, Name is a collection of FirstName and LastName, Address is a collection of Street, Town, City and Country.

### 4.2.1.3 Semantic Object Attributes

Semantic object attributes are the attributes the establish relationships between one semantic object to another semantic object.

### Example

Following is a simple example that explains the use of semantic object diagram. The objects in this diagram are indicated by rectangles. The name of the object appears on the top of the rectangle and the attributes are written after the name of the object in the rectangle.

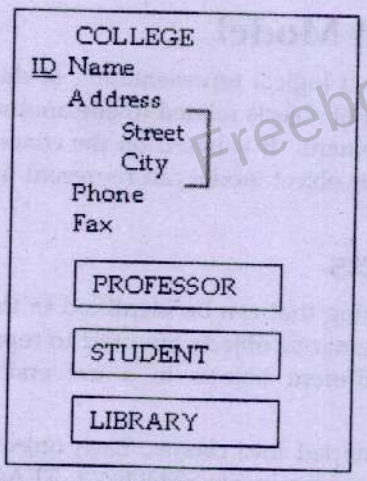



**Figure:** COLLEGE Object Diagram

The COLLEGE object diagram contains different attributes. Professor, Student and Library are semantic object attributes. The semantic object attributes indicate that COLLEGE object is logically connected with other objects. It means that Professor, Student and Library are also some other semantic objects.

## 4.2.2 Attribute Cardinality

Each semantic object contains two types of cardinalities:

- **Minimum Cardinality:** The minimum cardinality indicates the minimum number of instances of the attribute that must exist.
- **Maximum Cardinality:** The maximum cardinality indicates the maximum number of instances of the attribute that may exist.

Each attribute has both minimum and maximum cardinality. The minimum cardinality is usually 0 or 1. The value 0 indicates that the attribute is not compulsory. If it is 1, the attribute must have a value. Sometimes, minimum cardinality is more than 1. For example, PLAYERS attribute of CRICKET-TEAM object should have a minimum cardinality 11 as the cricket team cannot have less than 11 players.

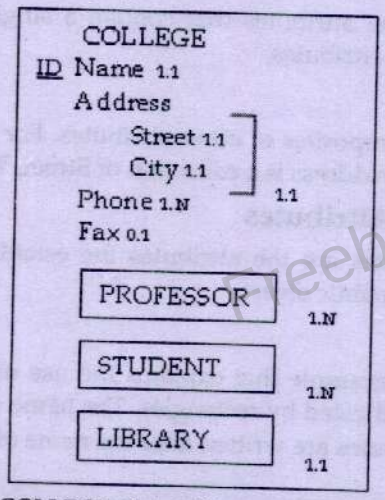



**Figure:** COLLEGE Object Diagram with cardinality

The maximum cardinality is usually 1 or N. If it is 1, the attribute cannot have more than 1 instance. If it is N, the attribute can have any number of instances. Sometimes, maximum cardinality is given as an exact value. For example, PLAYERS attribute of CRICKET-TEAM object should have a maximum cardinality 11 as the cricket team cannot have more than 11 players.

Cardinalities are specified in **n.m** format where **n** indicates the minimum cardinality and **m** indicates the maximum cardinality. In the above diagram, Name attribute of COLLEGE object has a minimum cardinality 1 and maximum cardinality 1. It means that COLLEGE object must have exactly one name. Phone attributes has minimum cardinality 1 and maximum cardinality N. It means that COLLEGE must have at least one phone but can have many phones.

## 4.2.3 Object Class & Object Instance

An **object class** is a general format for all object instances of that particular type. The previous figure shows the structure of a college and can be used for any college. When we represent an object instance, the figure will contain the values of all attributes. Following is an example of a COLLEGE **object instance**.

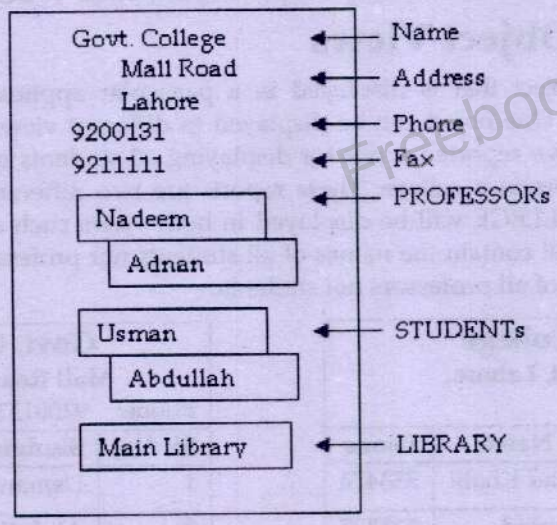



**Figure:** Object instance of COLLEGE

## 4.2.4 Object Identifiers

An **object identifier** is an attribute or collection of attributes that are used to identify an object instance. For example, possible identifiers of STUDENT are Roll No or Name. All attributes are not identifiers because some attributes cannot be used to identify a particular object instance. For example, Marks of a student is not an identifier because you cannot identify a student by using marks.

A **group identifier** is a type of identifier that has more than one attribute. For example, you can identify a student by using an identifier that consists of FirstName and SecondName.

**Object identifier** can be unique or non-unique. For example, Roll No of a student is a unique identifier but Name is not unique. There may possibly be two students with a name "Usman". In this situation, "Usman" identifies a group of students and one student will be identified by another attribute like Roll No etc.

In semantic diagram, object identifiers are indicated by the letters "ID" in front of the attribute. If the identifier is unique, "ID" is underlined.

## 4.3 Attribute Domains

A collection of possible values of an attribute is known as **attribute domain**. The domain of a simple attribute consists of two types of descriptions:

- Physical Description
- Semantic Description

The **physical description** indicates the type of data. For example, Name of a student should be a string value and Roll No should not exceed 99 etc.

The **semantic description** indicates the function or purpose of the attribute. For example, the Name of a student should be a valid name. This is a semantic description of Name attribute of the student. This restriction is related to the meaning of the attribute. For example, "Math" is not a valid name even if it consists of string value because it does not satisfy the semantic description of Name attribute.

Sometimes, the physical description is specified in a predefined list of values. For example, Class attribute of a student may be specified as ["MCS", "BCS", "MBA", "MS"].

## 4.4 Semantic Object Views

A part of an object that is displayed in a particular application is called **view** or **semantic object view**. One object can be displayed in different views as required. Suppose we want to generate two reports i.e. one for displaying all students in a college and one for displaying all professors in a college. These reports are two different views of COLLEGE. Some attributes of COLLEGE will be displayed in both views such as Name and Address. But the first reports will contain the names of all students not professors. The second report will contain the names of all professors not students.

**Govt. College**
**Mall Road, Lahore.**
Phone: 9200131

| Sr. No. | Professor Name | Phone |
|---|---|---|
| 1 | Muhammad Khalil | 733474 |
| 2 | Ashfaq Shahid | 721327 |
| 3 | Ahmad Kamal | 733233 |
| 4 | Ehsan Qadir | 721327 |
| 5 | Abid Ali | 733233 |

**Govt. College**
**Mall Road, Lahore.**
Phone: 9200131

| Sr. No. | Student Name | Marks |
|---|---|---|
| 1 | Usman Khalil | 835 |
| 2 | Abdullah | 712 |
| 3 | Ali | 584 |
| 4 | Nauman Qadir | 637 |
| 5 | Nadeem | 886 |

**Figure:** Two different views of COLLEGE object

## 4.5 Types of Objects

Types of objects use some new terms, which are as follows:

- **Single-Valued Attribute** – It is an attribute whose maximum cardinality is 1.
- **Multi-Valued Attribute** – It is an attribute whose maximum cardinality is greater than 1.
- **Non-object Attribute** – It is a simple or group attribute.

Different types of objects are as follows:

## 4.5.1 Simple Objects

A simple object is a semantic object that contains only single-valued, non-object attributes. For example, the following figure shows a simple object COMPUTER. This object contains only simple attributes. So this object is called a simple object.

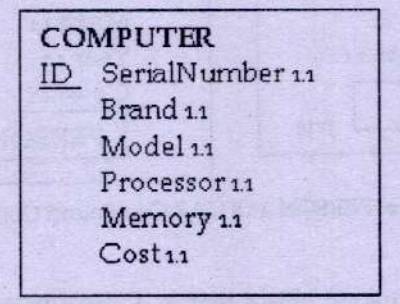



**Figure:** Simple Object COMPUTER

## 4.5.2 Composite Objects

A composite object is a semantic object that contains one or more multi-valued, non-object attributes. For example, the following figure shows a composite object BILL. This object contains a multi-valued attribute Services. A customer may get one or more services. So this is called a composite object.

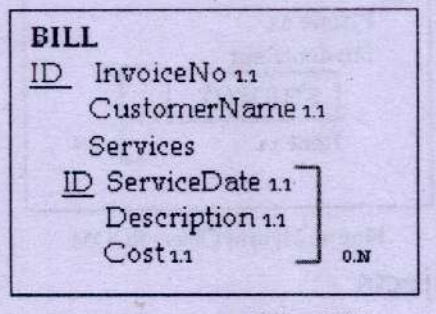



**Figure:** Composite Object BILL

## 4.5.3 Compound Objects

A compound object is a semantic object that contains at least one object attribute. For example, the following figure shows a compound object BOOK. This object contains an object attribute Author. So this is called a composite object.

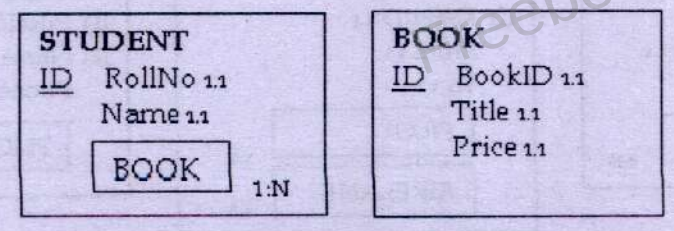



**Figure:** Compound Object STUDENT

In the above figure, STUDENT is a compound object because it contains an object attribute BOOK. But the BOOK object is a composite object as discussed in previous section. In some situations, it is possible that first object contains an object attribute of the second object and second object contains an object attribute of first object as follows:

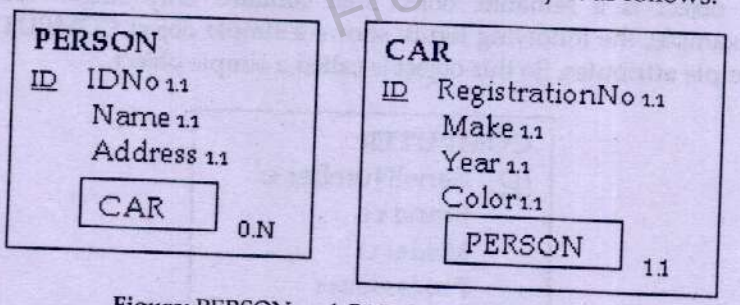



**Figure:** PERSON and CAR Compound Objects

## 4.5.4 Hybrid Objects

A hybrid object is a semantic object that is a combination of two objects types. For example, if an object contains the properties of both composite and compound types, it will be called a hybrid object. The following figure shows a hybrid object ROOM. This object contains a multi-valued attribute StudentRent that consists of STUDENT object attribute and Rent attribute. So this is called a hybrid object.

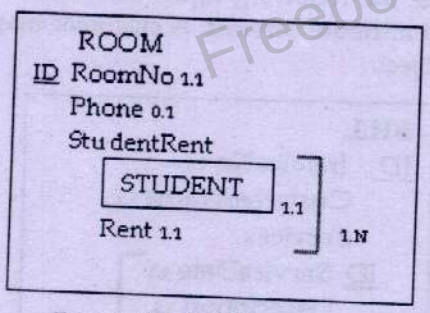



**Figure:** Hybrid Object ROOM

## 4.5.5 Association Objects

An association object is a semantic object that connects two or more objects and stores data about that relationship. The following example shows three objects PILOT, AIRPLANE and FLIGHT. The FLIGHT object is used as association object and stores data about the association of PILOT and AIRPLANE.


**Figure:** Association Object

## 4.5.6 Parent / Subtype Objects

A subtype object is a semantic object that inherits the attributes of an existing object. The inherited object is called parent object. Suppose we have an object PERSON with different attributes such as Name, Address, and Phone etc. If we want to create another object STUDENT, we can inherit the attributes of PERSON object in STUDENT object. Because STUDENT is also a PERSON and has same attributes like PERSON object like Name, Address, and Phone.

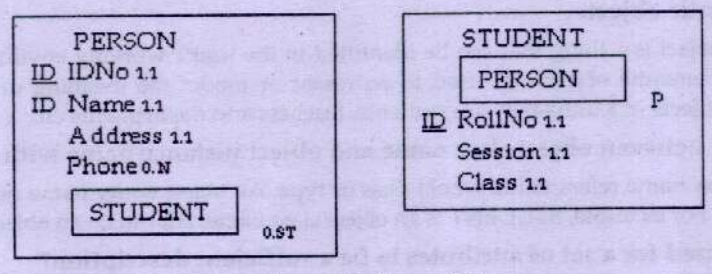



**Figure:** Parent/Subtype Objects

In the above figure, PERSON is the parent object and STUDENT is subtype object. The cardinality of STUDENT attribute in PERSON object is given as "0.ST". It indicates that a person can be a student. If it is 1, then it becomes necessary for each PERSON object to contain an object attribute STUDENT.

A parent object can have more than one subtype objects. For example, we can have an EMPLOYEE subtype object of PERSON parent object as follows:

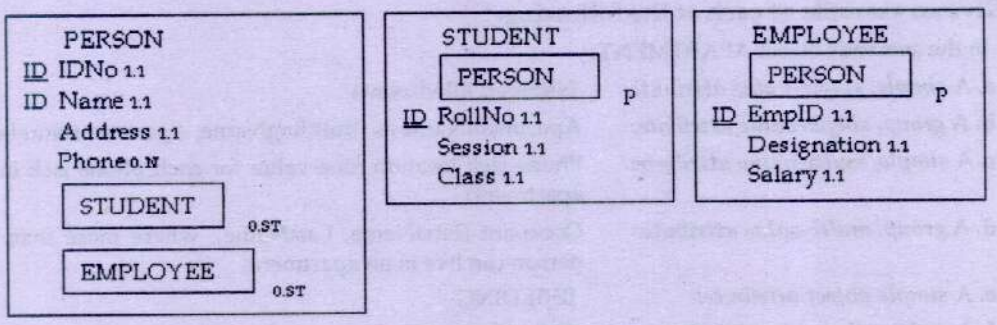



**Figure:** Parent/Subtype Objects

## 4.5.7 Archetype / Version Objects

An archetype object is a semantic object that produces other objects to represent versions, releases or editions of the archetype object. Suppose we have a semantic object BOOK with some attributes. We can create another object to represent different editions of the book as follows:

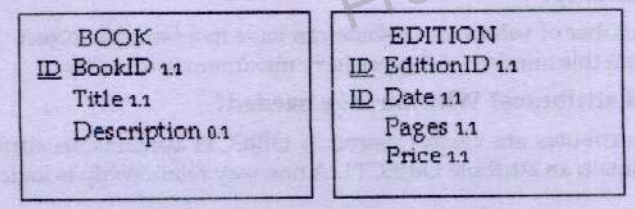



**Figure:** Archetype/Version Objects

## Short Questions

**Q.1. Explain why the E-R model and the semantic object model are like lenses.**

These models are tools for representing and expressing the views of users' data structures. They shape the image of the representation that is seen and documented.

**Q.2. Define semantic object.**

A semantic object is a thing that can be identified in the user's working environment. Semantic means "meaning". Semantic objects are used to represent or model the meaning of user's data. For example, different objects in a university are students, teachers and departments etc.

**Q.3. Differentiate between object class name and object instance name with example.**

An object class name refers to the whole class or type. An object entity name refers to a member of a particular class. For example, STUDENT is an object class name. Ali can be an object instance name.

**Q.4. What is required for a set of attributes to be a sufficient description?**

The properties represent all of the characteristics that the users need to perform their work.

**Q.5. Explain the words distinct identity as they pertain to definition of a semantic object.**

They mean something that users recognize as independent and separate. That thing stands on its own in the users' minds. Each instance of an object is unique and identifiable in its own right. Users have names for them such as Order-number or Employee-number.

**Q.6. List the three types of attributes**

Three types of attributes are Simple attributes, group attributes and semantic object attributes.

**Q.7. Give an example of each of the following:**

In the semantic object APARTMENT:

| | |
|---|---|
| a. A *simple, single-value attribute:* | NumberOfBedrooms |
| b. A *group, single-value attribute:* | ApartmentName as (BuildingName, ApartmentNumber) |
| c. A *simple, multi-value attribute:* | Phone jack location (one value for each phone jack in the apartment) |
| d. A *group, multi-value attribute:* | Occupant (FirstName, LastName), where more than one person can live in an apartment. |
| e. A *simple object attribute:* | BUILDING |
| f. *A multi-value object attribute:* | REPAIR |

**Q.8. What is minimum cardinality? How is it used? Which types of attributes have minimum cardinality?**

The minimum number of values of an attribute that are required is called minimum cardinality. No object is allowed to exist for which this number is not satisfied. All types have minimum cardinality.

**Q.9. What is maximum cardinality? How is it used? Which types of attributes have maximum cardinality?**

The maximum number of values an attribute can have in a semantic object. No object is allowed to exist that has more than this number. All types have maximum cardinality.

**Q.10. What are paired attributes? Why are they needed?**

Semantic object attributes are always paired. If OBJECT1 contains an attribute OBJECT2, then OBJECT2 will always contain an attribute OBJECT1. A one-way relationship is logically impossible.

**Q.11. What is an object identifier? Give an example of a simple attribute object identifier and an example of a group attribute object identifier.**

An object identifier is an attribute or collection of attributes that are used to identify an object instance. In AUTO, LicenseNumber is a simple attribute identifier. In APARTMENT, (BuildingName, ApartmentNumber) is a composite identifier

**Q.12. Define attribute domain. What are the types of attribute domain? Why is a semantic description necessary?**

A collection of possible values of an attribute is known as attribute domain. The domain of a simple attribute consists of two types of descriptions:

1. Physical Description

The physical description indicates the type of data. For example, Name of a student should be a string value and Roll No should not exceed 99 etc.

2. Semantic Description

The semantic description indicates the function or purpose of attribute. For example, Name of a student should be a valid name. It is a semantic description of Name attribute. This restriction is related to the meaning of the attribute. For example, "Math" is not a valid name even if it consists of string value because it does not satisfy the semantic description of Name attribute.

**Q.13. What is a semantic object view? Give an example of an object and two views other than those in this text.**

A semantic object view is a subset of a semantic object. Consider the object APARTMENT with attributes BuildingName, ApartmentNumber, RentAmount, LastOccupiedDate. One view might have the first three attributes and another might have all four attributes.

**Q.14. Give an example of a simple object**

SOFTWARE with properties Name, Type, Price, MemoryRequired.

**Q.15. Give three examples of composite objects. One example should have one multi-value simple attribute; one should have two independent multi-value groups and third should have nested multi-value groups.**

a. EMPLOYEE with properties Emp#, $\{\text{ReviewDate, ReviewComments}\}_{0.N}$.

b. EMPLOYEE with properties Emp#, $\{\text{ReviewDate, ReviewComments}\}_{0.N}$, and $\{\text{SalaryRevisionDate, Salary}\}_{0.N}$.

c. EMPLOYEE with properties Emp#, $\{\text{ReviewDate, }\{\text{ReviewerName, ReviewerComments}\}_{0.N}\}_{0.N}$. For last example, there are multiple reviews on a given date.

**Q.16. Give an example of four sets of compound objects. One set should have a 1:1 relationship, one set should have a 1:N relationship, one set should have an M:1 relationship and one set should have an M:N relationship.**

| | |
|---|---|
| 1:1 | COMPUTER contains EMPLOYEE |
| | EMPLOYEE contains COMPUTER |
| 1:N | PROJECT contains COMPUTERs |
| | COMPUTER contains only one PROJECT |
| M:1 | (really same as 1:N, which is the point of this part of the question) |
| N:M | COMPUTER contains SOFTWARE-PACKAGEs |
| | SOFTWARE-PACKAGE contains COMPUTERs |

**Q.17. Give an example of a hybrid object.**

COMPUTER contains the multi-valued group {SOFTWARE-PACKAGE, Price} 0.N, where SOFTWARE-PACKAGE is an object and Price is not.

**Q.18. Give an example of one association and two compound objects.**

JOB contains a single value of the object properties ARTIST and CUSTOMER.

CUSTOMER contains many values of JOB

ARTIST contains many values of JOB

**Q.19. Give an example of a supertype object with three subtype objects.**

SOFTWARE contains the properties Name, Price, Vendor and the subtype objects WORD-PROCESSING-SOFTWARE, SPREADSHEET-SOFTWARE and DBMS-SOFTWARE. WORD-PROCESSING-SOFTWARE contains the properties Name, Fonts0.N. SPREADSHEET-SOFTWARE contains the properties Name, GraphStyles0.N. DBMS-SOFTWARE contains the properties Name, MaxTables, MaxColumns, MaxRows.

**Q.20. Explain the similarities between E-R model and the semantic object model.**

Both are tools for understanding and documenting users' data models. Both are concerned with representing the things that are important to the users and the relationships among those things.

**Q.21. Explain the major differences between the E-R model and the semantic object model.**

E-R model takes entity as the basic element of interest to the user. Semantic object model takes semantic object as the basic element. Semantic objects show relationships in context of objects containing other objects. E-R model shows relationships as between entities. Usually, an entity is a different name for a table. Entities normally do not have composite attributes nor do they have multi-valued attributes. Entities do not contain other entities.

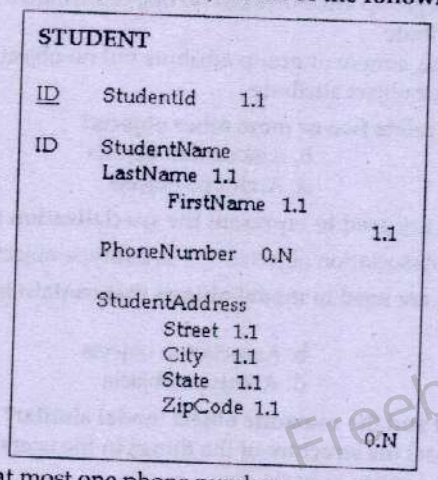
---

## Multiple Choice

1. **The term 'semantic' means:**
   a. Data b. Meaning c. Attribute d. Detailed
2. **Which type of attribute is composed of other attributes?**
   a. Simple b. Meta c. Group d. Compound
3. **Which of the following is not true of semantic objects?**
   a. Always represent physical entities b. Always provide a sufficient description
   c. Are always named d. Always describe a distinct identity
4. **A semantic object:**
   a. Is a representation of some identifiable thing in the users' work environment
   b. Is a characteristic of an attribute?
   c. Is one or more object attributes that the users employ to identify object instances
   d. Is description of an attribute's possible values?
5. **A semantic object that contains one or more multi-value, simple or group attributes but no other attributes is known as:**
   a. Simple objects b. Composite objects c. Compound objects d. Hybrid objects
6. **Combinations of composite and compound objects are known as:**
   a. Hybrid objects b. Association objects
   c. Subtype objects d. Archtype objects
7. **A semantic object that contains only single-value, simple or group attributes is called:**
   a. Simple objects b. Composite objects
   c. Compound objects d. Hybrid objects

8. **Which of the following contains at least one object attribute?**
   a. Simple objects b. Composite objects
   c. Compound objects d. Hybrid objects
9. **A hybrid object is a semantic object that contains at least one:**
   a. Single-value, group attribute that includes a multi-value semantic object attribute
   b. Single-value, group attribute that includes any semantic object attribute
   c. Multi-value, group attribute and at least one multi-value semantic object attribute
   d. Multi-value, group attribute that includes any semantic object attribute
10. **A composite object is a semantic object that contains:**
   a. Single-value, simple or group attributes but no object attributes
   b. At least one object attribute
   c. At least one multi-value, simple or group attribute but no object attributes
   d. At least one multi-value object attribute
11. **Which of the following relate two or more other objects?**
   a. Hybrid objects b. Association objects
   c. Subtype objects d. Archtype objects
12. **Which of the following are used to represent the specialization of objects?**
   a. Hybrid objects b. Association objects c. Subtype objects d. Archtype objects
13. **Which of the following are used to model objects that contain base data along with multiple variations, or versions?**
   a. Hybrid objects b. Association objects
   c. Subtype objects d. Archtype objects
14. **How are the E-R model and the semantic object model similar?**
   a. Neither strives to model the structure of the things in the users' world
   b. Both see the concept of entity as basic
   c. They both see the semantic object as basic
   d. Both are tools for understanding and documenting the structure of -the users' data
15. **The principle difference between the E-R model and the semantic model is:**
   a. The E-R model sees the concept of entity as basic while the semantic object model sees the concept of semantic object as basic
   b. The semantic object model sees the concept of entity as basic while the E-R model sees the concept of semantic object as basic
   c. The E-R model models the users' world. The semantic model models the real world
   d. An E-R model is a tool for understanding and documenting the structure of the users' data while the semantic model is not
16. **Attributes which have a single element are called :**
   a. Simple attributes b. Group attributes
   c. Semantic object attributes d. Paired attributes
17. **One or more object attributes that the users employ to identify object instances is/are called:**
   a. Group identifiers b. Object identifiers c. Compound identifier d. Domain
18. **Which of the following is a description of an attribute's possible values?**
   a. Group identifiers b. Object identifiers c. Compound identifier d. Domain
19. **Which of the following is an identifier that has more then one attribute?**
   a. Group identifiers b. Object identifiers c. Compound identifier d. Domain
20. **The set of an attribute's specific values is called:**
   a. Identifier b. Domain c. Enumerated list d. Attribute
21. **Which of the following refers to an attribute whose maximum cardinality is 1?**
   a. Multi-value attribute b. Nonobject attribute.
   c. Paired attribute. d. Single-value attribute

22. **Which of the following refers to a simple or group attribute?**
    a. Multi-value attribute
    b. Nonobject attribute
    c. Paired attribute.
    d. Single-value attribute
23. **Which are the three types of domains?**
    a. Group, compound, and association
    b. Simple, group, and semantic
    c. Compound, hybrid, and association
    d. Simple, compound and semantic
24. **The process of developing a set of object diagrams is:**
    a. Sequential
    b Iterative
    c. Parallel
    d. Singular
25. **In the STUDENT semantic object above, which of the following is true?**




    a. a student may have at most one phone number
    b. a student may or may not have a value for zip code
    c. a student must have at least one phone number
    d. a student can have at most one street
    e. a student may or may not have a Student Id
26. **In the STUDENT semantic object above, which of the following is not true?**
    a. a student must have a Student Id
    b. a student must have exactly one last name
    c. a student must have exactly one City
    d. a student may not have a phone number
    e. a student may have many phone numbers
27. **A group identifier:**
    a. has more than one attribute
    b. identifies a group of instances
    c. identifies a group of attributes in an object
    d. is the identifier for a group of semantic objects

## Answers

| | | | |
|---|---|---|---|
| 1. b | 2. b | 3. a | 4. a |
| 5. b | 6. a | 7. a | 8. c |
| 9. d | 10. c | 11. b | 12. c |
| 13. d | 14. d | 15. a | 16. a |
| 17. b | 18. d | 19. a | 20. c |
| 21. d | 22. b | 23. b | 24. b |
| 25. b | 26. c | 27. a | |

## True / False

1. The domain of an attribute is a description of an attribute's possible values.
2. A composite object is a semantic object that contains one or more multi-value, simple or group attributes, but no object attributes.
3. A simple object is a semantic object that contains only single-value, simple or group attributes.
4. A compound object contains two or more object attributes.
5. Hybrid objects are combinations of composite and compound objects.
6. An association object is an object that relates two (or more) objects and stores data that are peculiar to that relationship.
7. A subtype object never inherits the attributes from its parent.
8. An archtype object is a semantic object that produces other semantic objects that represent versions, releases, or editions of the archtype.
9. A semantic object is a named collection of attributes that sufficiently describes a distinct identity.
10. Object attributes are always paired.
11. Subtype objects are used to represent the specializations of objects.
12. Both the E-R model and the semantic object model consider entities as basic.
13. The semantic model contains more information about the meaning of the data than the E-R model does.
14. A semantic object model is not a data model.
15. The word semantic means syntax.
16. Both entities and objects are similar in that they both have a collection of attributes.
17. A collection of attributes is a sufficient description if the attributes represent all of the characteristics that the users need in order to do their work.
18. Group attributes are composites of other attributes.
19. An object identifier is a single attribute that the user employs to identify an object instance.

## Answers

| | | | |
|---|---|---|---|
| 1. T | 2. T | 3. T | 4. F |
| 5. T | 6. T | 7. F | 8. T |
| 9. T | 10. T | 11. T | 12. F |
| 13. T | 14. F | 15. F | 16. T |
| 17. T | 18. T | 19. F | |

CHAPTER 5

# Relational Model & Normalization

## Chapter Overview

# 5.1 Relational Model

Dr. E. F. Codd worked to improve the working of DBMSs to handle large volumes of data. He applied the rules of mathematics to solve the problems of earlier database models. Some important problems were as follows:

- Data Integrity
- Data Redundancy

**Dr. Codd** presented a paper "A Relational Model of Data for Large Shared Databanks" in June 1970 that contained 12 rules. A DBMS that satisfies these rules is called a **full Relational Database Management System (RDBMS)**. The term **relation** is also derived from the **set theory** of mathematics.

In a relational model, data is stored in relations. Relation is another term used for table. A table in a database has a unique name that identifies its contents. Each table can be called an intersection of rows and columns. An important property of a table is that the rows are unordered. A row cannot be identified by its position in the table. Every table must have a column that uniquely identifies each row in the table.

## 5.1.1 Relation Database Terminology

Some important terminologies used in relational database model are as follows:

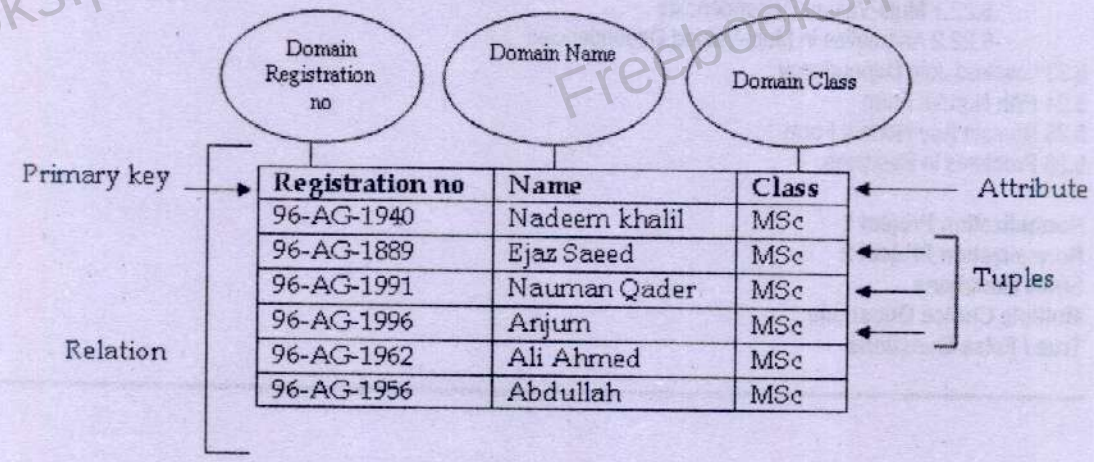



**Figure:** Database Terminology

### Relation

In a relational model, data is stored in relations. Relation is another term used for table. Following is an example of a relation.

| RegistrationNo | Name | Class |
|---|---|---|
| 96-AG-1940 | Nadeem Khalil | MSc |
| 96-AG-1889 | Ejaz Saeed | MSc |
| 96-AG-1991 | Nauman Qader | MSc |

**Figure:** An example of Relation

### Tuple

In a relational model, every relation or table consists of many tuples. Tuples are also called records or rows.

| | | |
|---|---|---|
| 96-AG-1940 | Nadeem Khalil | MSc |

| | | |
|---|---|---|
| 96-AG-1991 | Nauman Qader | MSc |

**Figure:** Two tuples of a Relation

### Attributes

An attribute is a named column of a relation. Attributes are also called characteristics. The characteristics of the tuple are represented by attributes or fields.

| |
|---|
| 96-AG-1940 |

| |
|---|
| Ejaz Saeed |

| |
|---|
| MSc |

**Figure:** Three attributes of a Relation

### Domain

A domain is a collection of all possible values of one or more attributes. For example, the value in the field "Class" can be the name of any taught classes. It is known as class domain. Similarly, Registration domain is a collection of all possible Registration numbers.

### Degree

The number of attributes is called the degree of that relation.

### Cardinality

The number of rows is called the cardinality of that relation.

## 5.1.2 Advantages of a Relational Database Model

Some important advantages of a relational database model are as follows:

### 1. Data Integrity

Relational model allows data integrity from field level to table level to avoid duplication of records. It detects records with missing primary key values at the relationship level to ensure valid relationships between relations.

### 2. Data Independence

The implementation of database will not be affected by changes made in the logical design of the database or changes made in the database software.

### 3. Structural Independence

**Structural independence** exists when the structure of database can be changed without affecting DBMS's ability to access the data. The relational database model does not use a navigational data access system. The data access paths are irrelevant to relational database designers, programmers and end users. Any change in relational database structure does not affect data access in any way. It makes relational databases model **structure independence.**

### 4. Data Consistency & Accuracy

Since multiple level check and constraints are built-in, data is accurate and consistent.

### 5. Easy Data Retrieval & Sharing

Data can be easily extracted from one or multiple relations. Data can also be easily shared among users.

## 5.2 Keys

A **key** is an attribute or set of attributes that uniquely identifies a tuple in a relation. The keys are defined in tables to access or sequence the stored data quickly and smoothly. They are also used to create relationship between different tables.

### 5.2.1 Super Key

A **super key** is an attribute or combination of attributes in a relation that identifies a tuple uniquely within the relation. A super key is the most general type of key.

For example, a relation STUDENT consists of different attributes like RegistrationNo, Name, FatherName, Class and Address. The only attribule that can uniquely identify a tuple in the relation is RegistrationNo. The Name attribute cannot identify a tuple because two or more students may have the same name. Similarly, FatherName, Class and Address cannot be used to identify a tuple. It means that RegistrationNo is the super key for the relation.

Any combination of attribues with the super key is also a super key. It means any attribute or set of attributes combined with the super key RegistrationNo will also become a super key. A combination of two attributes {RegistrationNo, Name} is also a super key. This combination can also be used to identify a tuple in the relation. Similarly, {RegistrationNo, Class} or {RegistrationNo, Name, Class} are also super keys.

### 5.2.2 Candidate Key

A **candidate key** is a super key that contains no extra attribute. It consists of minimum possible attributes. A super key like **{RegistrationNo, Name}** contains an extra field **Name**. It can be used to identify a tuple uniquely in the relation. But it does not consist of minimum possible attribute as only **RegistrationNo** can be used to identify a tuple in relation. It means that {RegistrationNo, Name} is a super key but it is not a candidate key because it contains an extra field. On the other hand, RegistrationNo is a super key as well as a candidate key.

### 5.2.3 Primary Key

A **primary key** is a candidate key that is selected by the database designer to identify tuples uniquely in a relation. A relation may contain many candidate keys. When the designer selects one of them to identify a tuple in the relation, it becomes a primary key. It means that if there is only one candidate key, it will be automatically selected as primary key.

Some most important points about a primary key are:

- A relation can have only one primary key.
- Each value in primary key attribute must be unique.
- Primary key cannot contain null values.

Suppose a relation Student contains different attributes such as RegNo, Name and Class. The attribute RegNo uniquely identifies each student in the table. It can be used as primary key for this table. The attribute Name cannot uniquely identify each row because two students can have same name. It cannot be used as primary key.

Primary Key

| RegNo | Name | Class |
|---|---|---|
| 10 | Nadeem Khalil | MSc |
| 20 | Muhammad Usman | BSc |
| 30 | Noman Qadir | FSc |

**Figure:** Primary Key

## 5.2.4 Alternate Key

The candidate keys that are not selected as primary key are known as **alternate keys**. Suppose Student relation contains different attributes such as RegNo, RollNo, Name and Class. The attributes RegNo and RollNo can be used to identify each student in the table. If RegNo is selected as primary key then RollNo attribute is known as alternate key.

Primary Key Alternate Key

| RegNo | RollNo | Name | Class |
|---|---|---|---|
| 10 | 1 | Nadeem Khalil | MSc |
| 20 | 2 | Muhammad Usman | BSc |
| 30 | 3 | Noman Qadir | FSc |

**Figure:** Alternate Key

## 5.2.5 Composite Key

A primary key that consists of two or more attributes is known as **composite key**. For example, the following relation uses two fields RollNo and Subject to identify each tuple. This is an example of composite key.

Composite Primary Key

| RollNo | Subject | Marks |
|---|---|---|
| 1 | English | 52 |
| 1 | Math | 77 |
| 1 | Computer | 64 |
| 2 | English | 58 |
| 2 | Math | 69 |
| 2 | Computer | 49 |
| 3 | English | 82 |
| 3 | Math | 98 |
| 3 | Computer | 86 |

**Figure:** Marks Table with composite key

## 5.2.6 Foreign Key

A **foreign key** is an attribute or set of attributes in a relation whose values match a primary key in another relation. The relation in which foreign key is created is known as **dependent table** or **child table**. The relation to which the foreign key refers is known as **parent table**. The key connects to another relation when a relationship is established between two relations. A relation may contain many foreign keys.

The following figure shows two relations. The RollNo attribute in Parent relation is used as primary key. The RollNo attribute in Child relation is used as foreign key. It refers to RollNo attribute in Parent relation.

Primary Key

| RollNo | Name | Address |
|---|---|---|
| 1 | Ali | Faisalabad |
| 2 | Ahmad | Lahore |
| 3 | Nadeem | Peshawar |

Parent Table

Composite Primary Key

| RollNo | Subject | Marks |
|---|---|---|
| 1 | English | 52 |
| 1 | Math | 77 |
| 1 | Computer | 64 |
| 2 | English | 58 |
| 2 | Math | 69 |
| 2 | Computer | 49 |
| 3 | English | 82 |
| 3 | Math | 98 |
| 3 | Computer | 86 |

Child Table

**Figure:** Foreign Key

## 5.3 Relational Database Management System

A **relational database management system (RDBMS)** is a DBMS that is based on the **relational model**. A relational database management system is a collection of software programs for creating, maintaining, modifying and manipulating a relational database. An RDBMS that satisfies the 12 rules of Dr. Codd is called a true RDBMS.

### 5.3.1 Components of RDBMS

A relational database system comprises of several components. Each component has a special role in the functioning of overall system. These components are also called the functional components. They are as follows:

#### 1. File Manager

File manage allocates space on the disk. It also manages the way data is organized and represented in storage.

#### 2. Database Manager

Database manager acts as an interface between the users and the data in the database.

#### 3. Query Processor

Most RDBMS provide built-in support to query the database. The query languages supported by most commercial DBMS are **Query By Example (QBE)** and **Structured Query Language (SQL)**. The query processor interprets the queries issued by the database users. A **Query Optimizer** also supports the optimization of complex queries involving nested queries or joins. The query processor translates statements in a query language to a form that can be understood by the database manager.

#### 4. Data Dictionary

The data dictionary stores information about data. It stores the description of the structure of the relation within a database. It also stores the description of data relationships and the integrity constraints on data.

### 5. DML Precompiler

DML stands for **Database Manipulation Language**. Database manipulation language has statements to insert, delete and modify data in database. The DML precompiler interprets these statements and interacts with query processor to generate appropriate code.

### 6. DDL Compiler

The **Data Definition Language** statements are converted to a set of files that contain information about the data. This is the system database that contains the following:

- Information about the files created
- Which fields they contain
- Users who have access rights to this file etc.

This information is stored in data dictionary that which holds the information about the database structure.

## 5.4 Types of Relations

Different types of relations in relational database system are as follows:

- Base Tables
- Query Results
- Views

### 5.4.1 Base Tables

A **base table** is a table that exists in a database. It is a table that is created by the user. It is not derived from another table. A base table can be created, altered and removed from a database. These tasks are accomplished using SQL statements.

### 5.4.2 Query Results

**Query** is a question in which the user asks the database management system to perform different operations on tables. For example, a user may wish to see all students who got an A grade. When a question is asked and query is executed, the resultant data is also stored in tables. Such tables are called query result tables.

### 5.4.3 Views

A **view** is a virtual table. Suppose a user wants to view a few columns of a base table instead of all columns. A view can be created to fulfill this request. The view consists of only those columns of the base table that the user requires. This format can be saved as a view by giving it a name.

Base Table →

| RegistrationNo | RollNo | Name | Class |
|---|---|---|---|
| 96-AG-1940 | 1 | Nadeem Khalil | MSc |
| 96-AG-1889 | 2 | Ejaz Saeed | MSc |
| 96-AG-1991 | 3 | Nauman Qader | MSc |

View →

| RegistrationNo | Name |
|---|---|
| 96-AG-1940 | Nadeem Khalil |
| 96-AG-1889 | Ejaz Saeed |
| 96-AG-1991 | Nauman Qader |

**Figure:** Base table & view

### 5.4.3.1 Advantages of Views

Some important advantages of using views are as follows:

#### 1. Security

Each user can be given permission to access the database only through a small set of views. The views contain the specific data the user is authorized to see. The user is restricted to access only specific portion of stored data.

#### 2. Query Simplicity

A view can draw data from several different tables and present it as a single table. The user can easily apply queries on this view as a single-table.

#### 3. Structural Simplicity

Views can give "personalized" view of the database to a user. It presents the database as a set of virtual tables that is more useful for a particular user.

#### 4. Insulation from Change

A view can present a consistent and unchanged image of the structure of the database even if the underlying source tables are split or restructured.

## 5.5 Properties of Relations

Relations have several properties. These properties are as follows:

#### 1. Atomic Values in Fields

An entry at the intersection of each row and column is atomic. There can be no multi-valued attributes or repeating groups in a relation.

#### 2. Entries from Same Domain

A **domain** is the type and range of values of attributes. In a relation, all entries in a given column belong to the same domain. For example, all entries in RegistrationNo attribute of a relation must be from RegistrationNo domain.

#### 3. Unique Tuples

Each tuple in a relation must be unique. For example, the RegistrationNo in each tuple of the table must be different from all other tuples. Uniqueness in a relation is guaranteed by assigning a primary key for each relation.

#### 4. Unique Attribute Name

The name of each attribute of a relation should be unique. A relation cannot have two identical attributes.

#### 5. Insignificant Attribute Sequence

The sequence of attributes in a relation is insignificant. This sequence can be changed without changing the meaning or use of the relation.

#### 6. Insignificant Tuple Sequence

The sequence of tuples in a relation is also insignificant. The sequence may be changed. If a new tuple is inserted in a relation, it is immaterial whether it is inserted at the beginning, at the end or in the middle of the relation.

## 5.6 Codd's Rules

The twelve rules of Dr. E.F. Codd for relational databases are as follows:

### Rule 1: The Information Rule

This rule requires that all information should be represented as data values in the rows and columns of tables. This is the basis of the relational model.

### Rule 2: The Guaranteed Access Rule

Every data value in a relational database should be accessible logically by specifying the relation name, primary key value and the attribute name.

### Rule 3: Systematic Treatment of NULL Values

DBMS must support NULL values to represent missing or inapplicable information. Null value must be distinct from zero or spaces. The NULL values must be independent of data type. It means that NULL values for all types are the same.

### Rule 4: Active Online Catalog Based on the Relational Model

The system catalog is a collection of relations that the DBMS maintains for its own use. These relations hold the description of the structure of database. These relations are created, owned and maintained by DBMS. The users can access them in the same manner as ordinary relations, depending on the user's privileges.

### Rule 5: Comprehensive Data Sub-language Rule

This rule states that the system must support at least all of the following functions:

- Data Definition
- View Definition
- Data Manipulation operations
- Security and Integrity Constraints
- Transaction Management operations

### Rule 6: View Updating Rule

All views that are theoretically updateable must be updateable by the system.

### Rule 7: High level Insert, Update and Delete

The rows should be treated as sets during insert, update and delete operations. The operations that modify the database should deal with sets and not only with single rows. Therefore, the DBMS must allow multiple rows to be updated.

### Rule 8: Physical Data Independence Rule

Application programs must remain unchanged when any changes are made in storage representation or access methods.

### Rule 9: Logical Data Independence Rule

The changes that do not modify any data stored in that relation, do not require changes to be made to application program.

### Rule 10: Integrity Independence Rule

Integrity constraints must be specified independent of application programs. They must be stored in the catalog.

### Rule 11: Distribution Independence Rule

Existing applications should operate successfully when the data is distributed. The physical location of data, the control program and the application may be different but the distribution should not affect the functioning of the application.

### Rule 12: Non-Subversion Rule

Database security or integrity cannot be bypassed. Every DBMS supports some security or integrity features and users must not be permitted to access any part of database by bypassing the security or integrity features. Therefore, integrity constraints should be specified at DBMS level and not through application programs only. The DBMS must ensure that no other level can bypass the constraints specified to maintain the integrity of database.

## 5.7 Relational Data Integrity

Data integrity means reliability and accuracy of data. Integrity rules are designed to keep the data consistent and correct. These rules act like a check on the incoming data. It is very important that a database maintains the quality of the data stored in it. DBMS provides several mechanisms to enforce integrity of the data in a column.

Enforcing data integrity ensures the quality of data in the database. For example, if an employee id is entered as "123", this value should not be entered again. The ID should not be assigned to two or more employees. Similarly if grades of students can be from A to F, the database should not accept any other value.

Data integrity falls into following categories:

- Entity integrity
- Domain integrity
- Referential Integrity

### 5.7.1 Entity Integrity

The entity integrity rule ensures that the primary key cannot contain null data. It is also called **row integrity**. If primary key is allowed to have null value, it is not possible to uniquely identify a tuple in relation. Entity integrity means that it should be easy to identify each entity in the database. An entity is any thing like an object, subject or event represented in the database. For example, a database for patient tracking in a hospital may have the following entities:

- Patients
- Prescriptions
- Physicians
- Drugs
- Appointments

A relation is created to store an entity. Its attributes represent the characteristics of an entity. Entity integrity defines a tuple as unique entity for a particular relation.

### 5.7.2 Domain Integrity

A set of values that can be stored in a column is called a **domain**. For example, the marks of a student in a subject can be from 0 to 100. **Domain integrity** enforces restrictions on the values entered in a column. It specifies the validity of a specific data entry in a column. The data type of a column enforces domain integrity. For example, if the data type of Experience column is numeric, it cannot store a value like "Two".

### 5.7.3 Referential Integrity

**Referential integrity** preserves the defined relationship between tables when records are added or deleted. It ensures that key values are consistent across the tables.

Primary Key

| RegistrationNo | Name | Class |
|---|---|---|
| 96-AG-1940 | Nadeem Khalil | MSc |
| 96-AG-1889 | Ejaz Saeed | MSc |
| 96-AG-1991 | Nauman Qader | MSc |

**Table:** Master Table

Foreign Key Primary Key

| RegistrationNo | Subject | Marks |
|---|---|---|
| 96-AG-1940 | OOP | 82 |
| 96-AG-1940 | DBMS | 77 |
| 96-AG-1940 | HTML | 91 |
| 96-AG-1889 | OOP | 74 |
| 96-AG-1889 | DBMS | 88 |
| 96-AG-1889 | HTML | 69 |
| 96-AG-1991 | OOP | 78 |
| 96-AG-1991 | DBMS | 95 |
| 96-AG-1991 | HTML | 80 |

**Table:** Child Table

**Figure:** Referential integrity

A value cannot be inserted in foreign key if it has no corresponding value in primary key field of the relation table. Such consistency requires that if a key value changes then all references to it should also be changed accordingly.

The value in foreign key column in the referencing table must be same as the value in the corresponding primary key column in referenced table. In the above figure, Child table is connected to Master table with RegistrationNo. The values of RegistrationNo in Child table must also be present in Master table.

The above example has two tables **Master** and **Child**. A value cannot be entered in Child table before entering the corresponding value in Master table. If the user wants to enter the result of a student with RegistrationNo "96-AG-1940", he has to enter the record in the Master table first. Then he can enter the details of that student in the Child table. Similarly, if a record is to be deleted from Master table, it is necessary to delete the corresponding records in Child table first.

## 5.8 Database Languages

A data sublanguage consists of two parts:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)

DDL is used to specify database schema. DML is used to read and update the database. These languages are called **data sublanguages** becase they do not provide constructs for all computing needs like conditional or iterative statements. Many DBMSs provide the facility to embed the sublanguage in a high-level programming language like COBOL, Fortran, C++, Java and Visual Basic etc. In this situation, the high-level language is called **host language**.

### 5.8.1 Data Definition Language

A language that is used to describe and name the entities, attributes, relationships, associated integrity and security constraints is called **data definition language**. DDL is used to express a set of definitions for specifying database schema. It is used to define or modify a schema. It is not used to manipulate data.

The result of compilation of DDL statements is a set of tables stored in a special file called **system catalog, data dictionary** or **data directory**. Data dictionary is a file that contains metadata. Metadata is the data about data and contains definitions of records, data items and other objects. Data dictionary is consulted before actual data are read or modified in the database system.

### 5.8.2 Data Manipulation Language

A language that supports the basic data manipulation operations on data in databases is called **data manipulation language**. Data manipulation operations include the following:

- Insertion of new data in database
- Modification of data in database
- Retrieval of data from database
- Deletion of data from database

DML applies to external, conceptual and internal levels. It is necessary to define effiecient low-level procedures to access data efficiently. On the other hand, ease of use is more important at higher levels of abstraction. The basic goal is to provide efficient human interaction with the system. A part of DML that is used to retrieve data is called **query language**. It is a high-level special-purpose language for retrieving data from the database.

There are basically two types of DML:

- **Procedural DML:** It requires a user to specify the required data and how to get the required data.
- **Nonprocedural DML:** It requires a user to specify the required without specifying how to get the data.

Nonprocedural DML is usually easier to learn and use than procedural DML. The user does not specify how to get data. These languages may generate code that is not as efficient as produced by procedural languages.

## 5.9 Relational Algebra

**Relational algebra** is a procedural query language that processes one or more relations to define another relation without changing original relations. The operands as well as the result are relations. The output of one operation can become the input of another operation to create nested expression in relational algebra. This property is known as **closure**.

### 5.9.1 Basic Operations of Relational Algebra

There are two categories of operations in relational algebra:

#### 1. Unary Operations

The operations which involve only one relation are called **unary operations**. The following operations are the unary operations:

- Selection
- Projection

#### 2. Binary Operations

The operations which involve pairs of relations are called **binary operations**. A binary operation uses two relations as input and produces a new relation as output. The following operations are the binary operations:

- Union
- Set Difference
- Cartesian Product

## 5.9.2 Selection Operation

The Selection operation is a unary operation. The selection operator is sigma $\sigma$. It acts like a filter on a relation. It returns only a certain number of tuples. It selects the tuples using a condition. The condition appears as subscript to $\sigma$. The resulting relation has the same degree as the original relation. However, the resulting relation may have fewer tuples than the original relation.

$\sigma_C$ (R) returns only those tuples in R that satisfy condition C

A condition C may consist of any combination of comparison or logical operators that operate on the attributes of R.

- Comparison operators: $=$ $<$ $>$ $\geq$ $\leq$ $\neq$
- Logical operators: $\wedge$ $\vee$ $\neg$
- Use the Truth tables for logical expressions:

and

| $\wedge$ | T | F |
|---|---|---|
| T | T | F |
| F | F | F |

or

| $\vee$ | T | F |
|---|---|---|
| T | T | T |
| F | T | F |

not

| $\neg$ | T | F |
|---|---|---|
| | F | T |

### Examples

Assume a relation EMP has the following tuples:

| Name | Office | Dept | Rank |
|---|---|---|---|
| Saleem | 400 | CS | Assistant |
| Junaid | 220 | Econ | Lecturer |
| Ghafoor | 160 | Econ | Assistant |
| Babar | 420 | CS | Associate |
| Saleem | 500 | Fin | Associate |

### Question

Select only those Employees who are in CS department:

$\sigma_{\text{Dept = 'CS'}}$ (EMP)

### Result

| Name | Office | Dept | Rank |
|---|---|---|---|
| Saleem | 400 | CS | Assistant |
| Babar | 420 | CS | Associate |

### Question

Select only those Employees with last name Saleem who are assistant professors:

$\sigma_{\text{Name = 'Saleem'}} \wedge_{\text{Rank = 'Assistant'}}$ (EMP)

### Result

| Name | Office | Dept | Rank |
|---|---|---|---|
| Saleem | 400 | CS | Assistant |

### Question

Select only those Employees who are Assistant Professors or in Economics department:

$$\sigma_{\text{Rank = 'Assistant'} \vee \text{Dept = 'Econ'}} (\text{EMP})$$

### Result

| Name | Office | Dept | Rank |
|---|---|---|---|
| Saleem | 400 | CS | Assistant |
| Junaid | 220 | Econ | Lecturer |
| Ghafoor | 160 | Econ | Assistant |

### Question

Select only those Employees who are not in the CS department or Lecturer:

$$\sigma_{\neg(\text{Rank = 'Lecturer'} \vee \text{Dept = 'CS'})} (\text{EMP})$$

### Result

| Name | Office | Dept | Rank |
|---|---|---|---|
| Ghafoor | 160 | Econ | Assistant |
| Saleem | 500 | Fin | Associate |

## 5.9.3 Projection Operator

Projection is also a unary operator. The Projection operator is pi $\pi$. It limits the attributes returned from the original relation. The resulting relation has the same number of tuples as the original relation. The degree of the resulting relation may be equal to or less than the original relation.

The general syntax is: $\pi_{\text{attributes}}$ R

Where attributes is the list of attributes to be displayed and R is the relation.

### Examples

Assume the same EMP relation used in previous examples.

### Question

Display only the names and departments of the employees:

$$\pi_{\text{name, dept}} (\text{EMP})$$

### Result

| Name | Dept |
|---|---|
| Saleem | CS |
| Junaid | Econ |
| Ghafoor | Econ |
| Babar | CS |
| Saleem | Fin |

### Combining Selection and Projection

The selection and projection operators can be combined to perform both operations.

### Question

Show the names of all employees working in the CS department:

$\pi_{name}$ ( $\sigma_{Dept = 'CS'}$ (EMP) )

### Result

| Name |
|---|
| Saleem |
| Babar |

### Question

Show name and rank of those Employees who are not in CS department or Lecturers:

$\pi_{name, rank}$ ( $\sigma_{\neg (Rank = 'Lecturer' \vee Dept = 'CS')}$ (EMP) )

### Result

| Name | Rank |
|---|---|
| Ghafoor | Assistant |
| Saleem | Associate |

## 5.9.4 Set Operations

The Selection and Projection operations extract information from one relation. The set operations are used to extract information from several relations. The relational algebra has set operations of Union, Set difference, Intersection and Cartesian product. These operations are also called **binary operations**.

### 5.9.4.1 Union

The union operation of two relations combines the tuples of both relations to produce a third relation. If two relations contain identical tuples, the duplicate tuples are eliminated. The notation for the union of two relations A and B is A UNION B. It is donated by U.

If A is defined as A = { a, b, c } and set B is defined as B = { a, c, 1, 2 }, then C = A U B will return C = { a, b, c, 1, 2 }.

The relations used in the union operation must have same number of attributes. The corresponding attributes must also come from same domain. Such relations are also called **union compatible** relations.

Union are commutative operations:

$A \cup B = B \cup A$

### Example: $A \cup B$

Following is an example of union operation. Two relations A and B are combined together by using union operator.

Table A

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 2 | B | 20 |
| 3 | C | 30 |

Table B

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 4 | D | 40 |
| 5 | E | 50 |

A UNION B

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 2 | B | 20 |
| 3 | C | 30 |
| 4 | D | 40 |
| 5 | E | 50 |

**Figure:** Union operation

### 5.9.4.2 Difference

The difference operation works on two relations. It produces a third relation that contains the tuples that occur in the first relation but not in second. The difference operation can be performed on union compatible relations. The order of subtraction is significant.

The difference operator is not commutative. It means:

$A - B \neq B - A$

### Example

Following is an example of difference operation. There are two relations A and B. The result of A – B and B – A are as follows:

Table A

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 2 | B | 20 |
| 3 | C | 30 |

Table B

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 4 | D | 40 |
| 5 | E | 50 |

A – B

| X | Y | Z |
|---|---|---|
| 2 | B | 20 |
| 3 | C | 30 |

B – A

| X | Y | Z |
|---|---|---|
| 4 | D | 40 |
| 5 | E | 50 |

**Figure:** The Difference operation

### 5.9.4.3 Intersection

The intersection operation works on two relations. It produces a third relation that only common tuples. Both relations must be union compatible. It is denoted by ∩.

### Example

Following is an example of intersection operation. There are two relations A and B. The result of A INTERSECTION B is as follows:

Table A

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 2 | B | 20 |
| 3 | C | 30 |

Table B

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 4 | D | 40 |
| 5 | E | 50 |

A ∩ B

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |

**Figure:** The Intersection operation

### 5.9.4.4 Product

The product works on two relations. It concatenates every tuple in one relation with every tuple in second relation. It is also called **cartesian product** or **cross product**. The product of relation A with **m** tuples and relation B with **n** tuples is relation C with m x n tuples. The product is denoted as A X B.

The Product needs not to be union compatible. It means that they can be of different degree. It is commutative and associative.

Table A

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 2 | B | 20 |
| 3 | C | 30 |

Table B

| X | Y | Z |
|---|---|---|
| 1 | A | 10 |
| 4 | D | 40 |
| 5 | E | 50 |

A X B

| X1 | Y1 | Z1 | X2 | Y2 | Z2 |
|---|---|---|---|---|---|
| 1 | A | 10 | 1 | A | 10 |
| 1 | A | 10 | 4 | D | 40 |
| 1 | A | 10 | 5 | E | 50 |
| 2 | B | 20 | 1 | A | 10 |
| 2 | B | 20 | 4 | D | 40 |
| 2 | B | 20 | 5 | E | 50 |
| 3 | C | 30 | 1 | A | 10 |
| 3 | C | 30 | 4 | D | 40 |
| 3 | C | 30 | 5 | E | 50 |

**Figure:** The Product operation

### 5.9.4.5 Division

The division operator results in columns values in one table for which there are other matching column values corresponding to every row in another table.

A

| K | X | Y |
|---|---|---|
| 10 | 1101 | A |
| 10 | 1201 | B |
| 10 | 1301 | C |
| 20 | 1201 | B |
| 30 | 1101 | A |
| 30 | 1201 | B |
| 30 | 1301 | C |

B (Divisor)

| X | Y |
|---|---|
| 1101 | A |
| 1201 | B |
| 1301 | C |

Result

| K |
|---|
| 10 |
| 30 |

**Figure:** Division operator

## 5.10 Join

A join operation combines the product, selection and possibly projection. The join operator combines data from one tuple of a relation with tuples from another or same relation when certain criteria are met. The criteria involve a relationship among the attributes in the join relations. Different types of joins are as follows:

- Theta Join
- Equi Join
- Natural Join
- Outer Join

### 5.10.1 Theta Join

Theta join is the result of performing a SELECT operation using a comparison operator **theta** on the product. Theta is denoted by θ.

In normal corss product, all rows of one relation are merged with all rows of second relation. In Theta join, only selected rows of first relation are merged with all rows of second relation. It is denoted as follows:

**R [X] S**

### Example

Suppose there are two relations FACULTY and COURSE as follows:

FACULTY Relation

| FacID | Name | Department | Designation |
|---|---|---|---|
| 1001 | Usman Khalil | CS | Assistant |
| 1002 | Abdullah | Fin | Associate |
| 1003 | Aysha Ashfaq | Eco | Associate |
| 1004 | Tanveer Hussain | Fin | Lecturer |

COURSE Relation

| CourseID | Title | FID |
|---|---|---|
| CS-520 | DBMS | 1001 |
| CS-511 | OOP | |
| CS-430 | FM | 1003 |

$(\sigma_{\text{Designation="Associate"}}(\text{FACULTY})) \times \text{COURSE}$

| FacID | Name | Department | Designation | CourseID | Title | FID |
|---|---|---|---|---|---|---|
| 1002 | Abdullah | Fin | Associate | CS-520 | DBMS | 1001 |
| 1002 | Abdullah | Fin | Associate | CS-511 | OOP | |
| 1002 | Abdullah | Fin | Associate | CS-430 | FM | 1003 |
| 1003 | Aysha Ashfaq | Eco | Associate | CS-520 | DBMS | 1001 |
| 1003 | Aysha Ashfaq | Eco | Associate | CS-511 | OOP | |
| 1003 | Aysha Ashfaq | Eco | Associate | CS-430 | FM | 1003 |

## 5.10.2 Equi Join

Equi join is a type of join in which tuples are joined on the basis of values of a common attribute between two relations. It is the most commonly used type of join. The common attributes of both relations appear twice in the output.

### Example

Apply the following operation on FACULTY and COURSE relations:

$\text{FACULTY} \bowtie_{\text{FACULTY.FacID = COURSE.FID}} \text{COURSE}$

| FacID | Name | Department | Designation | CourseID | Title | FID |
|---|---|---|---|---|---|---|
| 1001 | Usman Khalil | CS | Assistant | CS-520 | DBMS | 1001 |
| 1003 | Aysha Ashfaq | Eco | Associate | CS-430 | FM | 1003 |

## 5.10.3 Natural Join

Natural join is same as equi join. The difference is that the common attributes appear only once. A natural join removes the duplicate attributes. In most systems, a natural join requires that the attributes have same name to identify attributes to be used in the join. It may require a renaming mechanism.

### Example

FACULTY $\bowtie_{FacID, FID}$ COURSE

| FacID | Name | Department | Designation | CourseID | Title |
|---|---|---|---|---|---|
| 1001 | Usman Khalil | CS | Assistant | CS-520 | DBMS |
| 1003 | Aysha Ashfaq | Eco | Associate | CS-430 | FM |

## 5.10.4 Outer Join

In outer join, all tuples of left and right relations are part of output. All tuples of left relation which are not matched with right relation are left as Null. Similarly, all tuples of right relation which are not matched with left relation are left as Null.

There are three forms of outer join depending on the data to be kept:

- **Left Outer Join:** It is denoted by ⟕. It includes all tuples of left hand relation and includes only the matching tuples from right hand relation. The unmatched rows are represented as Null.
- **Right Outer Join:** It is denoted by ⟖. It includes all tuples of right hand relation and includes only the matching tuples from left hand relation. The unmatched rows are represented as Null.
- **Full Outer Join:** It is denoted by ⟗. It includes all tuples of left and right relations.

### Example

Suppose there are two relations PEOPLE and MENU as follows:

PEOPLE

| Name | Age | Food |
|---|---|---|
| Ali | 21 | Burger |
| Bilal | 24 | Pizza |
| Chohan | 23 | Beaf |
| Daud | 19 | Meat |

MENU

| Food | Day |
|---|---|
| Pizza | Monday |
| Burger | Tuesday |
| Chicken | Wednesday |
| Fish | Thursday |
| Rice | Friday |

### Left Outer Join

PEOPLE ⟕ MENU will display the following output:

| Name | Age | Food | Day |
|---|---|---|---|
| Ali | 21 | Burger | Tuesday |
| Bilal | 24 | Pizza | Monday |
| Chohan | 23 | Beaf | *NULL* |
| Daud | 19 | Meat | *NULL* |

### Right Outer Join

PEOPLE ⟖ MENU will display the following output:

| Name | Age | Food | Day |
|---|---|---|---|
| Bilal | 24 | Pizza | Monday |
| Ali | 21 | Burger | Tuesday |
| *NULL* | *NULL* | Chicken | Wednesday |
| *NULL* | *NULL* | Fish | Thursday |
| *NULL* | *NULL* | Rice | Friday |

### Full Outer Join

PEOPLE ⟗ MENU will display the following output:

| Name | Age | Food | Day |
|---|---|---|---|
| Ali | 21 | Burger | Tuesday |
| Bilal | 24 | Pizza | Monday |
| Chohan | 23 | Beaf | NULL |
| Daud | 19 | Meat | NULL |
| NULL | NULL | Chicken | Wednesday |
| NULL | NULL | Fish | Thursday |
| NULL | NULL | Rice | Friday |

## 5.10.5 Semi Join

Semi join takes the natural join of two relations then projects the attributes of first relation only. After join and matching the common attributes of both relations, only attributes of first relation are projected.

### Example

There are two relations FACULTY and COURSE as follows:

FACULTY Relation

| FacID | Name | Department | Designation |
|---|---|---|---|
| 1001 | Usman Khalil | CS | Assistant |
| 1002 | Abdullah | Fin | Associate |
| 1003 | Aysha Ashfaq | Eco | Associate |
| 1004 | Tanveer Hussain | Fin | Lecturer |

COURSE Relation

| CourseID | Title | FID |
|---|---|---|
| CS-520 | DBMS | 1001 |
| CS-511 | OOP | |
| CS-430 | FM | 1003 |

FACULTY ⋉ COURSE will display the following output:

| FacID | Name | Department | Designation |
|---|---|---|---|
| 1001 | Usman Khalil | CS | Assistant |
| 1003 | Aysha Ashfaq | Eco | Associate |

# 5.11 Relational Calculus

**Relational calculus** is a nonprocedural relational data manipulation language. It enables the user to specify only what data to be retrieved not how to retrieve it. It is not related to the calculus in mathematics. It takes its name from a branch of symbolic logic called **predicate calculus**. There are two forms of relational calculus:

- Tuple Oriented Relational Calculus
- Domain Oriented Relational Calculus

### 5.11.1 Tuple Oriented Relational Calculus

The tuple oriented relation calculus is primarily used to find relation tuples for which a predicate is true. Tuple variables are used for this purpose. A variable that takes only the tuples of relation or set of relations as its range of values is called **tuple variable**. The range of tuple variable is specified as follows:

RANGE OF S IS STUDENT

Where S is a tuple variable and STUDENT is its range. It means that S represents a tuple of STUDENT. It is expressed as follows:

{S|P(S)}

It means "find a set of all tuples S such that P(S) is true" where P is predicate condition.

#### Example

Suppose the range of tuple variable R is STUDENT:

{R|R.Marks>40}

The above statement will find a set of all tuples from STUDENT where the value of Marks attribute is more than 40.

### 5.11.2 Domain Oriented Relational Calculus

In domain oriented relational tuples, the variables take values from domains instead of tuples of relations. If P(X1,X2,...,Xn) is a predicate with variables X1, X2,...,Xn then

{<X1,X2,...,Xn>|P(X1,X2,...,Xn)}

means a set of all domain variables X1, X2,...,Xn for which the predicate P(X1,X2,...,Xn) is true. In domain oriented relational calculus, a **membership condition** is used to determine whether the values belong to a relation. The following expression evaluates to true if and only if there is a tuple in relation X with values x, y, z for its three attributes:

<x,y,z> ∈ X

## 5.12 Relational Algebra vs. Relational Calculus

The relational algebra and the relational calculus have the same expressive power. It means that all queries that can be formulated using relational algebra can also be formulated using relational calculus. It was first proved by **E. F. Codd** in 1972. The proof is based on an algorithm known as **Codd's reduction algorithm.** The algorithm states that an arbitrary expression of relational calculus can be reduced to a semantically equivalent expression of relational algebra.

It is sometimes said that languages based on relational calculus are higher level than languages based on relational algebra. This is because the algebra partially specifies the order of operations but the calculus leaves it to the compiler or interpreter to determine the most efficient order of evaluation.

## 5.13 Database Anomalies

Database anomalies are the problems in relations that occur due to redundancy in the relations. These anomalies affect the process of inserting, deleting and modifying data in the relations. Some important data may be lost if a relation is updated that contains database anomalies. It is important to remove these anomalies in order to perform different processing on the relations without any problem.

### Types of Anomalies

Different types of database anomalies are as follows:

### 1. Insertion Anomaly

The insertion anomaly occurs when a new record is inserted in the relation. In this anomaly, the user cannot insert a fact about an entity until he has an additional fact about another entity.

### 2. Deletion Anomaly

The deletion anomaly occurs when a record is deleted from the relation. In this anomaly, the deletion of facts about an entity automatically deletes the fact of another entity.

### 3. Modification Anomaly

The modification anomaly occurs when the record is updated in the relation. In this anomaly, the modification in the value of specific attribute requires modification in all records in which that value occurs.

## 5.14 Normalization

The process of producing a simpler and more reliable database structure is called **normalization**. It is used to create a suitable set of relations for storing data. This process works through different stages known as **normal forms**. These stages are 1NF, 2NF, 3NF and so on. Each normal form has certain requirements or condition. These conditions have to fulfilled to bring the database in that particular normal form. If a relation satisfies the conditions of a normal form, it is said to be in that normal form.

The task of database design starts with unnormalized set of relations. The process of normalization identifies and corrects the problems and complexities of database design. It produces a new set of relations. The new design is as free of processing problems as possible.

### 5.14.1 Purposes of Normalization

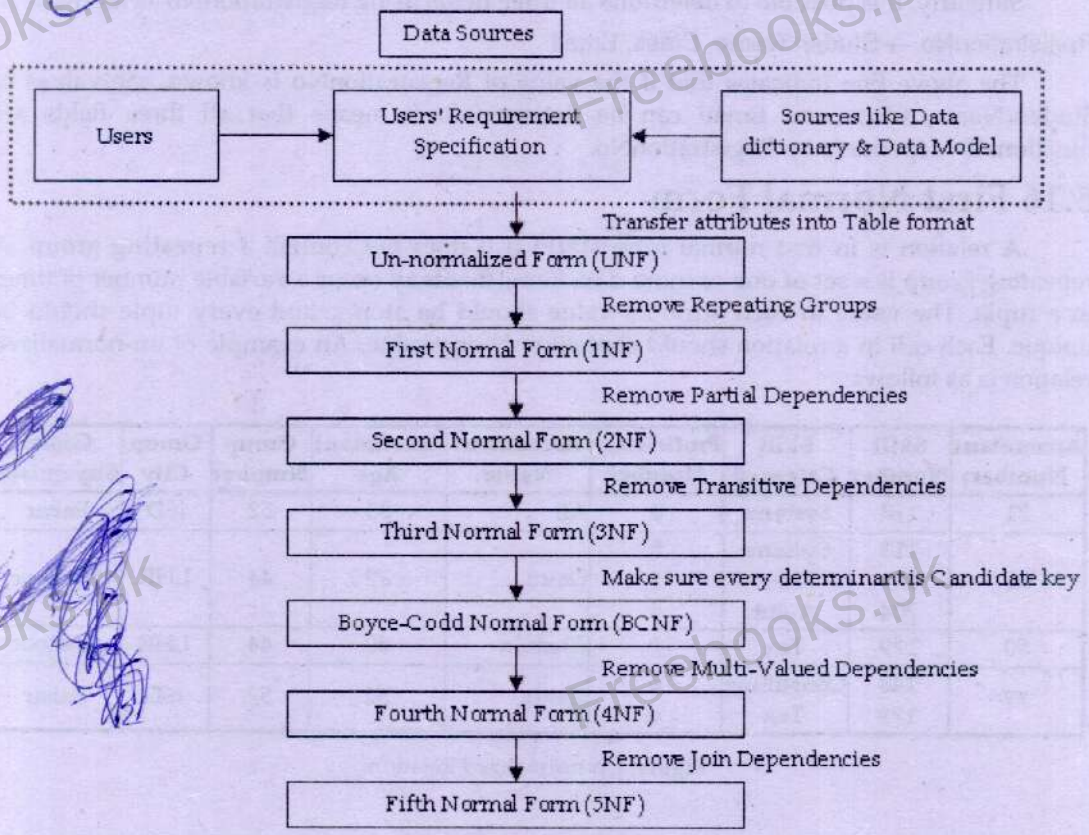
The purposes of normalization are as follows:

- It makes the database design efficient in performance.
- It reduces the amount of data if possible.
- It makes the database design free of update, insertion and deletion anomalies.
- It makes the design according to the rules of relational databases.
- It identifies relationship between entities.
- It makes a design that allows simple retrieval of data.
- It simplifies data maitenance and reduces the need to restructure data.

### 5.14.2 Characteristics of Normalized Database

A normalized database should have the following characteristics:

- Each relation must have a key field.
- All fields must contain atomic data.
- There must be no repeating fields.
- Each table must contain information about a single entity.
- Each field in a relation must depend on key fields.
- All non-key fields must be mutually independent.




## 5.15 Functional Dependency

Functional dependency is a relationship between attributes. It means that if the value of one attribute is known, it is possible to obtain the value of another attribute. Suppose there is a relation STUDENT with following fields:

STUDENT (RegistrationNo, StudentName, Class, Email)

If value of RegistrationNo is known, it is possible to obtain the value of StudentName. It means that StudentName is functionally dependent on RegistrationNo. An attribute B is functionally dependent on attribute A if the value of A determines the value of B.

Functional dependency is written as follows:

RegistrationNo → StudentName

The above expression is read as "RegistrationNo determines StudeName" or "StudentName is functionally dependent on RegistrationNo". The attribute on the left side is called **determinant**.

If A and B are attributes or sets of attributes of relation R, B is functionally dependent on A if each value of A in R has exactly one associated value of B in R.

A particular value of RegistrationNo is related to only one value of StudentName. But StudentName may be related with multiple values of RegistrationNo. For example, the RegistrationNo 10 is related with only one value of StudentName. But StudentName "Usman" may be related with two or more RegistrationNo values because two or more students may have the name "Usman".

Similarly, it is possible to determine all three fields using RegistrationNo as follows:

RegistrationNo → StudentName, Class, Email

The above line indicates that if the value of RegistrationNo is known, the values of StudentName, Class and Email can be determined. It means that all three fields are functionally dependent on RegistrationNo.

## 5.16 First Normal Form

A relation is in first normal form (1NF) if it does not contain a **repeating group**. A repeating group is a set of one or more data items that may occur a variable number of times in a tuple. The value in each attribute value should be atomic and every tuple should be unique. Each cell in a relation should contain only one value. An example of un-normalized relation is as follows:

| Accountant Number | Skill Number | Skill Category | Proficiency Number | Accountant Name | Accountant Age | Group Number | Group City | Group Supervisor |
|---|---|---|---|---|---|---|---|---|
| 21 | 113 | Systems | 3 | Ali | 55 | 52 | ISD | Babar |
| 35 | 113<br>179<br>204 | Systems<br>Tax<br>Audit | 5<br>1<br>6 | Daud | 32 | 44 | LHR | Ghafoor |
| 50 | 179 | Tax | 2 | Chohan | 40 | 44 | LHR | Ghafoor |
| 77 | 148<br>179 | Consulting<br>Tax | 6<br>6 | Zahid | 52 | 52 | ISD | Babar |

**Figure:** Un-normalized Relation

The above relation is un-normalized because it contains repeating groups of three attributes Skill Number, Skill Category and Proficiency Number. All three fields contain more than one value.

In order to convert this relation in first normal form, these repeating groups should be removed. The following relation is in first normal form:

**Primary Key**

| Accountant Number | Skill Number | Skill Category | Proficiency | Accountant Name | Accountant Age | Group Number | Group City | Group Supervisor |
|---|---|---|---|---|---|---|---|---|
| 21 | 113 | Systems | 3 | Ali | 55 | 52 | ISD | Babar |
| 35 | 113 | Systems | 5 | Daud | 32 | 44 | LHR | Ghafoor |
| 35 | 179 | Tax | 1 | Daud | 32 | 44 | LHR | Ghafoor |
| 35 | 204 | Audit | 6 | Daud | 32 | 44 | LHR | Ghafoor |
| 50 | 179 | Tax | 2 | Chohan | 40 | 44 | LHR | Ghafoor |
| 77 | 148 | Consulting | 6 | Zahid | 52 | 52 | ISD | Babar |
| 77 | 179 | Tax | 6 | Zahid | 52 | 52 | ISD | Babar |

**Figure:** First Normal Form (1NF)

### 5.16.1 Problems in 1NF

The relation in 1NF has certain problems which are as follows:

#### 1. Updating Problem

Suppose the user wants to change the name of Accountant Number 35 to "M. Daud". He has to the name in all records in which Accountant number 35 appears. This process of updating can be very lengthy.

#### 2. Inconsistent Data

The above table may contain inconsistent data. There are three records of Accountant Number 35. It is possible that there are two different names with Account Number 35 in two different records. The user can make this error during updating.

#### 3. Addition Problem

Suppose the user wants to add another skill number in the table. It is not possible until an Accountant with that skill exists because both Skill Number and Accountant Number are used as primary key in the above table.

#### 4. Deletion Problem

Suppose the user wants to delete the record of supervisor Ghafoor. If he deletes the whole record in which Ghafoor appears, the information about Accountants will also be lost.

## 5.17 Full Functional Dependency

In a relation R, attribute B of R is fully functionally dependent on an attribute or set of attributes A of R if B is functionally dependent on A but not functionally dependent on any other proper subset of A.

Suppose there is a relation MARKS as follows:

MARKS (RegistratioNo, Subject, Marks)

Assume that one student is studying many subjects. Both RegistrationNo and Subject are required to determine a particular marks. It can be written as follows:

RegistrationNo, Subject → Marks

Here, Marks is fully functionally dependent on both fields because it is not functionally dependent on either RegistrationNo or Subject alone.

## 5.18 Second Normal Form

A relation is in Second Normal Form (2NF) if it is in 1NF and if all of its nonkey attributes are fully functionally dependent on the whole key. It means that none of non-key attributes are related to a part of key.

The above relation in 1NF has some attributes which are not depending on the whole primary key. For example, Accountant Name, Accountant Age and group information is determined by Accountant Number and is not dependent on Skill. The following relation can be created in which all attributes are fully dependent on primary key Account Number.

Primary Key

| Accountant Number | Accountant Name | Accountant Age | Group Number | Group City | Group Supervisor |
|---|---|---|---|---|---|
| 21 | Ali | 55 | 52 | ISD | Babar |
| 35 | Daud | 32 | 44 | LHR | Ghafoor |
| 50 | Chohan | 40 | 44 | LHR | Ghafoor |
| 77 | Zahid | 52 | 52 | ISD | Babar |

**Figure:** Accountant Table in 2NF

Similarly, another relation Skill can be created in which all fields are fully dependent on the primary key as follows:

Primary Key

| Skill Number | Skill Category |
|---|---|
| 113 | Systems |
| 179 | Tax |
| 204 | Audit |
| 148 | Consulting |

**Figure:** Skill Table in 2NF

The attribute Proficiency in 1NF relation was fully dependent on the whole primary key. The Proficiency requires to know the accountant number and skill number. The third relation will be created as follows:

Primary Key

| Accountant Number | Skill Number | Proficiency |
|---|---|---|
| 21 | 113 | 3 |
| 35 | 113 | 5 |
| 35 | 179 | 1 |
| 35 | 204 | 6 |
| 50 | 179 | 2 |
| 77 | 148 | 6 |
| 77 | 179 | 6 |

**Figure:** Proficiency Table in 2NF

There are three relations in second normal form (2NF). The attributes of all relations are fully dependent on the primary keys.

### 5.18.1 Analysis of Second Normal Form (2NF)

The following analysis indicates whether the problems are eliminated in 2NF or not.

- **Updating Problem:** If the user needs to change the name of Accountant Number 35 to "M. Daud" in 1NF, he must change the name in every record in which Accountant number 35 appears. But in 2NF, the record of one accountant appears only once. The updating problem is eliminated in 2NF.
- **Inconsistent Data:** The record of one accountant appears only once in the database, the possibility of inconsistent data is automatically eliminated.
- **Addition Problem:** In 1NF, it was not possible to enter a new Skill Number until an Accountant with that skill existed. In 2NF, any number of skills can be added in Skill relation without an accountant with that skill. It eliminates the addition problem.
- **Deletion Problem:** In 2NF, if the record of Ghafoor is deleted, it does not delete any other record.

The analysis shows that the second normal form has solved all problems of 1NF.

## 5.19 Transitive Dependency

Transitive dependency is a condition in which an attribute is dependent on an attribute that is not part of the primary key.

Suppose A, B, and C are attributes of a relation. If A → B and B → C then C is transitively dependent on A via B provided that A is not functionally dependent on B or C.

Suppose there is a relation BOOK as follows:

BOOK (<u>BookID</u>, BookDescription, CategoryID, CategoryDescription)

We can write:

BookID → CategoryID, CategoryID → CategoryDescription

A transitive dependency occurs when one non-key attribute determines another non-key attribute. In above relation, BookID determines CategoryID and CategoryID determines CategoryDescription. CategoryDescription is transitively depedent on BookID attribute.

## 5.20 Third Normal Form

A relation is in third normal form if it is in 2NF and if no non-key attribute is dependent on another non-key attribute. It means that all non-key attributes are functionally dependent only on primary key. There should be no transitive dependency in a relation.

In order to convert a relation to 3NF:

- Remove all attributes from the 2NF record that depend on another non-key field
- Place them into a new relation with the other attribute as the primary key.

The Accountant table in 2NF contains some attributes which are depending on non-key attributes. For example, Group City and Group Supervisor are depending on a non-key field Group Number. A new relation can be created as follows:

Primary Key

| Accountant Number | Accountant Name | Accountant Age | Group Number |
|---|---|---|---|
| 21 | Ali | 55 | 52 |
| 35 | Daud | 32 | 44 |
| 50 | Chohan | 40 | 44 |
| 77 | Zahid | 52 | 52 |

**Figure:** Accountant Table in 3NF

The second table created form the Accountant table in 1NF is as follows:

Primary Key

| Group Number | Group City | Group Supervisor |
|---|---|---|
| 52 | ISD | Babar |
| 44 | LHR | Ghafoor |

**Figure:** Group table in 3NF

Both Accountant table and Group table contain the attribute Group Number. This attribute is used to join both tables.

The Skill table in 2NF contains no attribute, which is depending on a non-key attribute. It is already in third normal form and will be used without any further change.

Primary Key

| Skill Number | Skill Category |
|---|---|
| 113 | Systems |
| 179 | Tax |
| 204 | Audit |
| 148 | Consulting |

**Figure:** Skill Table in 3NF

The Proficiency table in 2NF also contains no attribute which is depending on non-key attribute. It is already in third normal form and will be used without any further change.

Primary Key

| Accountant Number | Skill Number | Proficiency |
|---|---|---|
| 21 | 113 | 3 |
| 35 | 113 | 5 |
| 35 | 179 | 1 |
| 35 | 204 | 6 |
| 50 | 179 | 2 |
| 77 | 148 | 6 |
| 77 | 179 | 6 |

**Figure:** Proficiency Table in 3NF

## 5.21 Boyce-Codd Normal Form (BCNF)

A relation is in **Boyce-Codd normal form** if and only if every determinant is a candidate key. It can be checked by identifying all determinants and then making sure that all these determinants are candidate keys. BCNF is a stronger form of third normal form. A relation in BCNF is also in third normal form. But a relation in 3NF may not be in BCNF.

Primary Key

| ProjectID | PartID | QuantityUsed | PartName |
|---|---|---|---|
| 101 | P01 | 20 | CD-R |
| 112 | P05 | 6 | Zip disk |
| 194 | P01 | 12 | CD-R |
| 194 | P02 | 1 | Box floppy disks |
| 194 | P05 | 3 | Zip disk |

**Figure:** PartsAndProject Table

Assume that PartName is unique. It means that no two parts can have the same name. There are now two candidate keys (ProjectID, PartID) and (ProjectID, PartName).

There are following dependencies:

(ProjectID, PartID) → QtyUsed
PartID → PartName
PartName → PartID

This relation satisfies 2NF because there are no non-key attributes that are dependent on a subset of the primary key. PartName is not a non-key attribute, it is part of a candidate key. Therefore this relation is in 2NF.

There are no transitive dependencies so the relation is in 3NF. PartID and PartName are ignored by 3NF rule because they are both key attributes.

In order to convert this relation to BCNF, all functional dependencies must be removed which have a determinant that is not a candidate key. The result is as follows:

Primary Key

| PartID | PartName |
|---|---|
| P01 | CD-R |
| P02 | Box floppy disks |
| P05 | Zip disk |

Primary Key

| ProjectID | PartID | QuantityUsed |
|---|---|---|
| 101 | P01 | 20 |
| 112 | P05 | 6 |
| 194 | P01 | 12 |
| 194 | P02 | 1 |
| 194 | P05 | 3 |

**Figure:** Tables in BCNF

## 5.22 Fourth Normal Form

A relation is in 4NF if it is in BCNF and has no multi-valued dependencies.

### 5.22.1 Multi-Valued Dependencies

A multi-valued dependency exists when a relation has at least three attributes, two of them are multi-valued and their values depend on only the third attribute.

Suppose there is a relation R(A, B, C). A multi-valued dependency exists if:

- A determines multiple values of B
- A determines multiple values of C
- B and C are independent of each other.

For example, the following relation has multi-valued dependencies:

| StudentID | Subject | Interest |
|---|---|---|
| 100 | Math | Reading |
| 100 | Accounting | Reading |
| 100 | Math | Tennis |
| 100 | Accounting | Tennis |
| 150 | Economics | Jogging |

**Figure:** Table with multi-valued Dependencies

In the above example, a student can study many subjects and can have many interests. There are four records for StudentID 100. This is because we have to use all possible combination of Subject and Interest to make it clear that a student with any subject can have any interest. Otherwise, it would appear that Reading can only occur with Math and Tennis can occur only with Accounting.

Multi-valued dependencies are written as follows:

StudentID → → Subject
StudentID → → Interest

## 5.22.2 Anomalies in Multi-Valued Dependencies

Suppose StudentID 100 decides to take another course "Economics". It requires the addition of two tuples i.e. one with combination of Reading and one with combination of Tennis as follows:

| | | |
|---|---|---|
| 100 | Economics | Reading |
| 100 | Economics | Tennis |

**Figure:** New tuples in table

Similarly, if a student wants to drop a subject, all tuples containing that particular subject have to be deleted.

In order to eliminate these anomalies, multi-valued dependency should be eliminated. It can be done by creating two relations. Each relation will contain data for only one multi-valued attribute. The resulting relations do not have anomalies.

| StudentID | Subject |
|---|---|
| 100 | Math |
| 100 | Accounting |
| 150 | Economics |

| StudentID | Interest |
|---|---|
| 100 | Reading |
| 100 | Tennis |
| 150 | Jogging |

**Figure:** Elimination of Multi-Valued Dependency

## 5.23 Lossless Join Dependency

A property of decomposition that ensures that no spurious tuples are generated when relations are reunited through a natural join operation.

A relation that is decomposed during normalization can be rejoined by using Joins to produce the original data. Sometimes, a relation is decomposed into more than two relations. Lossless join dependency ensures that if the decomposed relations are reunited, data is not lost and no additional tuple is generated. The real focus of lossless join dependency and fifth normal form are the cases where the relation is decomposed in more than two relations.

## 5.24 Fifth Normal Form

A relation is in fifth normal form if it has no join dependency. Fifth normal form (5NF) is also called **project-join normal form (PJNF)**. The first four normal forms are based on the concept of functional depdency. The fifth normal is based on the concept of join dependency.

## 5.25 Domain Key Normal Form

A relation is in DK/NF if every constraint on the relation is a logical consequence of the definition of keys and domains. It means that a relation is in DK/NF if enforcing key and domain restrictions causes all of the constraints to be met.

DK/NF is important because a relation in DK/NF has no modification anomalies. A relation with no modification anomalies must be in DK/NF. DK/NF involves only the concepts of key and domain. These concepts are fundamental in database environment. They are readily supported by DBMS products.

Suppose there is a relation STUDENT as follows:

STUDENT(SID, NAME, MAJOR, CREDITS)

Suppose a rules thates that SID should have prefix to indicate the type of student. The prefix 1 indicates freshman, 2 indicates sophomores and so on. It can be used to express a general constraint "if first digit of SID is 1 then CREDITS must be between 0 and 30 so on." The constraint must be expressible as domain constraint or key constraint to convert a relation in DKNF. The constraint can be expressed by splitting STUDENT trelation in four different relations. For example, the following relation can be used for freshman:

STD1(SID, NAME, MAJOR, CREDITS)

The above relation will have the following constraints:

- SID must begin with 1.
- CREDITS must be between 0 and 30.

Similarly, STD2 will be created for sophomores, STD3 for juniors and STD4 for seniors.

The basic concept of DKNF is simple but there is no proven method of converting a desing to this form. It remains an ideal rather than a state that can be achieved readily.

## 5.26 Problems in Relations

Different problems in the relations are as follows:

### 1. Synonym

A **synonym** is a type of problem that exists in relations. A synonym is created when two different names are used for the same information or attribute. The name of attribute must be same if it exists in two or more relations.

The following example displays two relations with synonym problem:

| ITEM | SUPPLIER |
|---|---|
| Stock_No | Supplier_ID |
| Item_Color | Supplier_Name |
| Supplier_Code | |

The above relations are inter-related. The ITEM relation contains information about the items that are supplied by the supplier. The SUPPLIER relation has an attribute Supplier_ID. The ITEM relation is referring to Supplier_ID with Supplier_Code that is wrong. It must also use Supplier_ID in order to refer to Supplier_ID attribute of SUPPLIER relation.

## 2. Homonym

A **homonym** is a type of problem that exists in relations. A homonym is created when same name is used for two different attributes. The following example displays two relations with homonym problem:

| CUSTOMER | SUPPLIER |
|---|---|
| Company_Name | Company_Name |

The attribute **Company_Name** is appearing in both relations. It may create confusion. The solution is that unique attribute names must be used in all relations to avoid confusion.

## 3. Redundancy

**Redundancy** means duplication of data in multiple files. It is a type of problem that exists in relations. It is created when the same information is unnecessarily stored in two ways or forms. The following example displays a relation with redundancy problem:

| EMPLOYEE |
|---|
| Date_of_Birth |
| Age |

The relation contains two attributes. The first attribute stores the date of birth of an employee. The second attribute stores the age of an employee. The age can be calculated by using the date of birth. It means that Age attribute is not necessary. It is creating redundancy in the relation and must be dropped from the relation.

## 4. Mutual Exclusiveness of Data

The data that does not have overlapping information is known as **mutually exclusive data**. The mutual exclusiveness of data creates problem in some cases. It creates problem for the attributes whose values can be specified as "Yes/No" form. Sometimes, two or more such attributes cannot be **true** or **false** at the same time for one entity. The following example displays a relation with this problem:

| EMPLOYEE |
|---|
| Married |
| Single |

The above two attributes cannot be **true** or **false** at the same. The problem occurs if "Yes" is selected in both attributes. The problem can be solved by using a larger categorical attribute. The above relation can have an attribute "MARITAL_STATUS". The possible values in this attribute can be "M" and "S" where "M" indicates "Married" and "S" indicates "Single".

| EMPLOYEE |
|---|
| Marital_Status |

## Normalization Project 1

The following table depicts the set of attributes found in a database:

| StdID | StdName | SocietyID | SocName | SupID | Supervisor | Position |
|---|---|---|---|---|---|---|
| 123 | Masood | 001 | Urdu | 1 | Mr. Waseem | Chairman |
| | | 003 | Maths | 2 | Ms. Chauhdry | Member |
| 132 | Khalida | 001 | Urdu | 1 | Mr. Waseem | Member |
| 142 | Javed | 002 | English | 1 | Mr. Waseem | Member |
| | | 005 | Physics | 3 | Mr. Liaqat | Chairman |
| | | 008 | Biology | 4 | Miss Yasmeen | Member |

**Figure:** Un-normalized Relation **Student_in_Society**

### Solution

The above table is in un-normalized form. We will apply first three normal forms on the above relation. The solution is as follows:

### First Normal Form

The **Student_in_Society** table is in un-normalized form as **SocietyID** is a multi-valued attribute. The above relation can be converted into 1NF by storing the details of the repeating groups in a separate table. This will result in the following table structures.

Student (StdID, StdName)
Student_in_Society (StdID, SocietyID, SocietyName, SupID, Superviser, Position)

| StdID | StdName |
|---|---|
| 123 | Masood |
| 132 | Khalida |
| 142 | Javed |

**Student** relation

| StdID | SocietyID | SocName | SupID | Supervisor | Position |
|---|---|---|---|---|---|
| 123 | 001 | Urdu | 1 | Mr. Waseem | Chairman |
| 123 | 003 | Maths | 2 | Ms. Chauhdry | Member |
| 132 | 001 | Urdu | 1 | Mr. Waseem | Member |
| 142 | 002 | English | 1 | Mr. Waseem | Member |
| 142 | 005 | Physics | 3 | Mr. Liaqat | Chairman |
| 142 | 008 | Biology | 4 | Miss Yasmeen | Member |

**Student_in_Society** relation

**Figure: Student** relation and **Student_in_Society** in 1NF

The above relations are in 1NF. However, many data anomalies still exist. Suppose that a new supervisor Mr. Khan replaces Mr. Waseem to become society teacher of Urdu Society. Two rows in **Student_in_Society** table need to be updated which is a modification anomaly. The relation also has an insertion anomaly. It is not possible to store information about a new society as no student has joined it. A deletion anomaly exists when the last member of a society quits. The society information will be permanently removed from the database.

### Second Normal Form

The partial dependencies are removed from the table to convert a table to 2NF. The functional dependencies for the **Student** table are as follows:

| | |
|---|---|
| StdID, SocietyID → Position: | Full functionally dependency |
| SocietyID → SocName: | Partial dependency as SocietyID is a part of primary key only |
| SocietyID → SupID: | Partial dependency as SocietyID is a part of primary key only |

The **Student_in_Society** table can be converted to 2NF by extracting **SocietyName**, **SupID**, **Superviser** to a separate table **Society**. These three attributes are fully functionally dependent on **SocietyID** which will be used as primary key in **Society** table as follows:

Student (StdID,StdName)
Society (SocietyID, SocietyName, SupID, Superviser)
Student_in_Society (StdID, SocietyID, Position)

| StdID | StdName |
|---|---|
| 123 | Masood |
| 132 | Khalida |
| 142 | Javed |

**Student** relation

| SocietyID | SocName | SupID | Supervisor |
|---|---|---|---|
| 001 | Urdu | 1 | Mr. Waseem |
| 002 | English | 1 | Mr. Waseem |
| 003 | Maths | 2 | Ms. Chauhdry |
| 005 | Physics | 3 | Mr. Liaqat |
| 008 | Biology | 4 | Miss Yasmeen |

**Society** relation

| StdID | SocietyID | Position |
|---|---|---|
| 123 | 001 | Chairman |
| 123 | 003 | Member |
| 132 | 001 | Member |
| 142 | 002 | Member |
| 142 | 005 | Chairman |
| 142 | 008 | Member |

**Student_in_Society** relation

**Figure: Student, Society** and **Student_in_Society** relations in 2NF

The above tables are in 2NF but are not able to solve all anomalies. The modification anomaly still exists in **Society** table. Suppose Mr. Waseem resigns and a new teacher Mr. Khan replaces him in all societies. Mr. Khan will use the same SupID of Mr. Waseem. Two rows instead of one need to be updated to reflect this change in **Society** table.

## Third Normal Form

A table is in 3NF if it is in 2NF and it exhibits no transitive dependencies. In the **Society** table, **SocietyID → SupID** and **SupID → Supervisor** and thus **SocietyID → Supervisor**. It is a kind of transitive dependency. The attributes that contribute to transitive dependencies are extracted to separate table(s) to convert a table to 3NF. The **Society** table can be converted to 3NF by extracting **Supervisor** to a new table **Society_Teacher**. The attribute **Supervisor** is fully functionally dependent on **SupID**. It is copied to **Society_Teacher** table to be used as primary key. This will result in the following table structures.

Student (StdID,StdName)
Society (SocietyID, SocietyName, SupID)
Student_in_Society (StdID, SocietyID, Position)
Society_Teacher (SupID, Superviser)

| StdID | StdName |
|---|---|
| 123 | Masood |
| 132 | Khalida |
| 142 | Javed |

**Student** relation

| SocietyID | SocName | SupID |
|---|---|---|
| 001 | Urdu | 1 |
| 002 | English | 1 |
| 003 | Maths | 2 |
| 005 | Physics | 3 |
| 008 | Biology | 4 |

**Society** relation

| StdID | SocietyID | Position |
|---|---|---|
| 123 | 001 | Chairman |
| 123 | 003 | Member |
| 132 | 001 | Member |
| 142 | 002 | Member |
| 142 | 005 | Chairman |
| 142 | 008 | Member |

**Student** relation

| SupID | Supervisor |
|---|---|
| 1 | Mr. Waseem |
| 2 | Ms. Chauhdry |
| 3 | Mr. Liaqat |
| 4 | Miss Yaseem |

**Society_Teacher** relation

**Figure: Student, Society, Student_in_Society** and **Society_Teacher** relations in 3NF

## Normalization Project 2

The following table depicts the set of attributes found in a University database:

| StudentID | StudentName | CourseID | CourseLength | UnitCode | UnitName | Lecturer |
|---|---|---|---|---|---|---|
| 001 | Usman | A203 | 3 | U45<br>U87 | Databases II<br>Programming | Ashraf<br>Ghafoor |
| 003 | Nadeem | A104 | 4 | U86<br>U45<br>U25 | Algorithms<br>Databases II<br>Business I | Naveed<br>Ashraf<br>Raoof |
| 007 | Abdullah | A203 | 3 | U12<br>U46 | Business II<br>Databases I | Abid<br>Zahid |
| 010 | Adnan | A323 | 2 | U12<br>U86 | Business II<br>Algorithms | Abid<br>Naveed |

**Figure:** Un-normalized Relation

Note that a student attends one course and can take any units during the course. A unit may be presented as part of any course and is always given by one particular lecturer.

### Solution

The above table contains un-normalized table. We will apply first three normal forms on the above relation. The solution is as follows:

### First Normal Form

To convert the above relation into first normal form (1NF), we need to remove all repeating groups. The set of attributes which repeat for each value of StudentID are UnitCode, UnitName, and Lecturer. In order to remove the repeating groups, we can use two approaches:

### First Approach

In the first approach, as used in the previous example, we enter data in the empty columns of rows that contain the repeating data. It removes repeating groups and now the relation contains atomic value in each field. In this approach, we get the problem of data redundancy that is solved in the next steps of normalization.

## Second Approach

One approach to do so is to split the relation in two relations to eliminate the repeating groups. We remove the repeating group by placing the repeating data along with a copy of original key attributes in a separate relation. A primary key is identified for the new relation. This approach produces relations with less redundancy. This example uses the second approach as follows:

Primary Key

| StudentID | StudentName | CourseID | CourseLength |
|---|---|---|---|
| 001 | Usman | A203 | 3 |
| 003 | Nadeem | A104 | 4 |
| 007 | Abdullah | A203 | 3 |
| 010 | Adnan | A323 | 2 |

Primary Key

| StudentID | UnitCode | UnitName | Lecturer |
|---|---|---|---|
| 001 | U45 | Databases II | Ashraf |
| 001 | U87 | Programming | Ghafoor |
| 003 | U86 | Algorithms | Naveed |
| 003 | U45 | Databases II | Ashraf |
| 003 | U25 | Business I | Raoof |
| 007 | U12 | Business II | Abid |
| 007 | U46 | Databases I | Zahid |
| 010 | U12 | Business II | Abid |
| 010 | U86 | Algorithms | Naveed |

**Figure:** Student table and Unit table in First Normal Form

## Second Normal Form

To convert the above relation into second normal form (2NF), we need to remove all partial dependencies. The Student relation does not have a composite primary key and, therefore, cannot contain partial dependencies. So, the Student is already in second normal form. The Unit relation has the following dependencies:

UnitCode → UnitName
UnitCode → Lecturer
StudentID, UnitCode → UnitName, Lecturer

Lecturer is only determined by UnitCode. Therefore, a partial dependency exists between UnitCode and UnitName and Lecturer. Removing the partial dependencies from Unit produces the following relations:

Primary Key

| UnitCode | UnitName | Lecturer |
|---|---|---|
| U45 | Databases II | Ashraf |
| U87 | Programming | Ghafoor |
| U86 | Algorithms | Naveed |
| U25 | Business I | Raoof |
| U12 | Business II | Abid |
| U46 | Databases I | Zahid |

Unit Table

Primary Key

| StudentID | UnitCode |
|---|---|
| 001 | U45 |
| 001 | U87 |
| 003 | U86 |
| 003 | U45 |
| 003 | U25 |
| 007 | U12 |
| 007 | U46 |
| 010 | U12 |
| 010 | U86 |

Take Table

**Figure:** Tables in Second Normal Form (2NF)

## Third Normal Form

To convert the above relation to third normal form (3NF), we need to remove all transitive dependencies. The Student table contains the following functional dependencies:

StudentID → StudentName, CourseCode, CourseLength
CourseCode → CourseLength

Therefore, the following transitive dependency exists:

StudentID → CourseCode → CourseLength

Removing this transitive dependency from Student produces the following relations:

Primary Key

| StudentID | StudentName | CourseID |
|---|---|---|
| 001 | Usman | A203 |
| 003 | Nadeem | A104 |
| 007 | Abdullah | A203 |
| 010 | Adnan | A323 |

Student Table

Primary Key

| CourseID | CourseLength |
|---|---|
| A203 | 3 |
| A104 | 4 |
| A323 | 2 |

Course Table

**Figure:** Tables in Third Normal Form (3NF)

The Take relation contains no non-key attributes and so contains no transitive dependencies. The Unit relation contains the following dependencies:

UnitCode → UnitName, Lecturer

Therefore, there are no transitive dependencies in Unit. The final relations in third normal form are as follows:

Primary Key

| StudentID | StudentName | CourseID |
|---|---|---|
| 001 | Usman | A203 |
| 003 | Nadeem | A104 |
| 007 | Abdullah | A203 |
| 010 | Adnan | A323 |

Student Table

Primary Key

| CourseID | CourseLength |
|---|---|
| A203 | 3 |
| A104 | 4 |
| A323 | 2 |

Course Table

Primary Key

| UnitCode | UnitName | Lecturer |
|---|---|---|
| U45 | Databases II | Ashraf |
| U87 | Programming | Ghafoor |
| U86 | Algorithms | Naveed |
| U25 | Business I | Raoof |
| U12 | Business II | Abid |
| U46 | Databases I | Zahid |

Unit Table

Primary Key

| StudentID | UnitCode |
|---|---|
| 001 | U45 |
| 001 | U87 |
| 003 | U86 |
| 003 | U45 |
| 003 | U25 |
| 007 | U12 |
| 007 | U46 |
| 010 | U12 |
| 010 | U86 |

Take Table

**Figure:** Final tables in Third Normal Form (3NF)

## Short Questions



**Q.1. Contrast the following:**

a. Candidate key and primary key
b. Functional dependency and transitive dependency
c. Foreign key and primary key

a. **Candidate Key and Primary Key:** A primary key is an attribute or combination of attributes that uniquely identifies a row in a relation. When a relation has more than one such attribute or combination of attributes, each is called a candidate key.

b. **Functional Dependency and Transitive Dependency:** A functional dependency is a relationship between any two attributes or two sets of attributes. A transitive dependency is a functional dependency between two or more non-key attributes.

c. **Foreign Key and Primary Key:** A primary key uniquely identifies each row in a relation. a foreign key is an attribute or set of attributes in a relation that reference a primary key in another table.

**Q.2. What is the difference between a candidate key and a superkey?**

A superkey may not be minimal. A candidate key is a minimal superkey. A superkey is minimal if it does not remain unique by removing a column.

**Q.3. What is the relationship between the referential integrity rule and foreign keys?**

Referential integrity involves the values that can be used in foreign keys. A foreign key can store a value matching a candidate key value in some row of the table containing the associated candidate key or a null value.

**Q.4. What is a primary key? Are duplicate primary keys allowed? Why or why not?**

A primary key is a field or set of fields that can uniquely identify a row of a table. Duplicate key values are not allowed because primary keys must uniquely identify a row.

**Q.5. What is a foreign key? Why are foreign keys used in a relational database? Are duplicate foreign key values allowed? Why or why not?**

A foreign key is a field or set of fields stored in one table that also exists as a primary key value in another table. Foreign keys are used to represent relationships among entities that are represented as tables. Duplicate foreign keys are not allowed within the same table because they would redundantly represent the same relationship. Duplicate foreign keys may exist in different tables because they would represent different relationships.

**Q.6. What are the difference between partial dependency and transitive dependency?**

A partial dependency exists when an attribute is dependent on only a part of primary key. It is associated with 1NF. Transitive dependency is a condition in which an attribute is dependent on another attribute that is not part of primary key. It usually requires the decomposition of the table containing the transitive dependency.

**Q.7. What restrictions must be placed on a table to consider it a relation?**

- Rows contain data about an entity
- Columns contain data about attributes of the entity
- Cells of the table hold a single value
- All entries in a column are of the same kind
- Each column has a unique name
- The order of the columns in unimportant
- The order of the rows in unimportant
- No two rows may be identical

**Q.8. Define relation, tuple, attribute, file, record, field, table, row and column.**

- A relation is a two-dimensional table that has the characteristics.
- A tuple is one row of a table.
- An attribute is one column in a table.
- A file is a term use by many people to a table.
- A record is the same as row of a table
- A field is the same as a column of a table.
- A table is a relation. Generally these terms are interchangeable.
- A row contains all data about one instance of an entity represented in a table.
- A column contains all values assigned to an attribute in a table.

**Q.9. What is a view? Discuss the difference between a view and a base relation.**

View is the dynamic result of one or more relational operations operating on the base relations to produce another relation. Base relation exists as a set of data in database. A view does not contain any data. It is defined as a query on one or more base relations. The query on view is translated into a query on the associated base relations.

**Q.10. Describe the purpose of normalizing data.**

The process of producing a simpler and more reliable database structure is called **normalization**. It is used to create a suitable set of relations for storing data. The process of normalization identifies and corrects the problems and complexities of database design. It produces a new set of relations. The new design is as free of processing problems as possible.

**Q.11. How is functional dependency associated with the process of normalization?**

Normalization is a technique to analyze relations based on primary key and functional dependencies. Normalization is often performed as a series of tests on a relation to determine whether it satisfies or violates the requirements of a given normal form. Three normal forms were initially proposed which are called 1NF, 2NF and 3NF. All of these normal forms are based on the functional dependencies among the attributes of a relation.

**Q.12. What is well-structured Relation? Why are well-structured relations important in logical database design?**

A well-structured relation is a relation that contains a minimum amount of redundancy. It allows users to insert, modify and delete rows in a table without errors or inconsistency. Well-structured relations are important because they promote database integrity.

**Q.13. Which three data anomalies are likely to be the result of data redundancy? How can such anomalies be eliminated?**

The most common anomalies considered when data redundancy exists are update anomalies, addition anomalies and deletion anomalies. All these can easily be avoided through normalization. Data redundancy produces data integrity problems.

**Q.14. What is a deletion anomaly? Give an example.**

A deletion anomaly is a case where deletion of facts about one entity instance deletes facts about another entity instance. The user can lose facts about two entities with one deletion.

**Q.15. What is an insertion anomaly? Give an example.**

An insertion anomaly is a case where the user cannot insert a fact about one entity until he has an additional fact about another entity.

**Q.16. What is the cause of modification anomalies?**

Poor database design causes the modification anomaly. A good database design avoids modification anomalies by eliminating excessive redundancies.

**Q.17. Describe the characteristics of an unnormalized table. How such table is converted to first normal form (1NF)?**

A table in unnormalized form contains one or more repeating groups. It is converted to first normal form by removing the repeating group to a new relation along with a copy of original key attribute(s). The repeating group can be removed by entering appropriate data in the empty columns of rows containing the repeating data.

**Q.18. What is a normal form?**

A normal form is a rule about allowable dependencies. Each normal form removes certain kinds of redundancies.

**Q.19. What does 1NF prohibit?**

1NF prohibits repeating groups in tables. A table not in 1NF is unnormalized or non-normalized.

**Q.20. What is a key column?**

A column is a key column if it is a candidate key or part of a candidate key.

**Q.21. What is a non-key column?**

A column is a non-key column if it is not a key column.

**Q.22. Define functional dependency. Give an example of two attributes with functional dependency and give an example of two attributes with no functional dependency.**

A functional dependency is a relationship between attributes such that given the value of one attribute it is possible to determine the value of the other attribute. Example of functional dependency: Name-->Phone#. Example of attributes that are not functionally dependent: Age and Address.

**Q.23. Which normal forms rely on the definition of functional dependency?**

Second and third normal forms are based on a concept called functional dependency—a one-to-one correspondence between two field values. Second normal form ensures that every field in a table is functionally dependent on the primary key. Third normal form ensures that no non-key field is functionally dependent on any other non-key field.

**Q.24. What is a view? How it is related to data independence?**

A view is a virtual table that does not really exist in its own. It is derived from one or more base table. There is no stored file that represents the view. A definition of view is stored in data dictionary. The view can insulate users from the effects of restructuring and growth in database. It provides logical data independence. .

**Q.25. Define determinant.**

A determinant is an attribute whose value enables us to obtain the value(s) of other related attributes. It appears on the left side of a functional dependency. Thus, in A--->B, the determinant is A.

**Q.26. What is removed when a relation is converted to the first normal form?**

All repeating groups are removed and the primary keys are identified when a relation is converted to the first normal form.

**Q.27. What is removed when a relation is converted from 1NF to 2NF?**

All the partially dependent attributes are removed and placed in another relation when a relation is converted from 1NF to 2NF.

**Q.28. What is difference between normal form and normalization?**

Normal form is a state of a particular relation regarding functional dependencies while the normalization is a process of producing a simpler and more reliable database structure.

**Q.29. What kinds of functional dependencies are not allowed in 2NF?**

Functional dependencies in which part of a key determines a nonkey column are not allowed in 2NF.

**Q.30. List three conditions that can be applied to determine that a relation in first normal form is also in Second Normal Form.**

Three conditions that imply a relation is in second normal form:

1. The primary key consists of a simple attribute.
2. No non-key attributes exist in the relation.
3. Every non-key attribute is functionally dependent on the full set of primary key attributes.

**Q.31. What is removed when a relation is converted from 2NF to 3NF?**

Any transitive dependencies, nonkey attributes dependent on other nonkey attributes, are removed when a relation is converted from 2NF to 3NF.

**Q.32. What kinds of functional dependencies are not allowed in 3NF?**

Functional dependencies in which a nonkey column determine another nonkey column are not allowed in 3NF.

**Q.33. What kinds of functional dependencies are not allowed in BCNF?**

Functional dependencies in which the determinant is not a candidate key are not allowed in BCNF..

**Q.34. Define second normal form. Give an example of a relation in 1NF but not in 2NF. Transform the relation into relations in 2NF.**

A relation is in second normal form if all its non-key attributes are dependent on the entire key. A table that is in second normal form cannot have a partial dependency.

Assume: COURSE (Dept_Code, Course_Number, Course_Title, Credits, Department_Name)

The table is in first normal form because it has no repeating attributes. Since the key is composite, it is not in second normal form if Dept_Code → Department_Name. Only part of primary key determines an attribute. To put the table in second normal form use the following tables.

COURSE (Dept_Code, Course_Number, Course_Title, Credits)

DEPARTMENT (Dept_Code, Department_Name)

**Q.35. Define third normal form. Give an example of a relation in 2NF but not in 3NF. Transform the relation into relations in 3NF.**

A relation is in third normal form if it is in second normal form and has no transitive dependencies. All determinants must be keys.

Assume: STUDENT (SIDU, Stu_Name, P_O_Box, Major, Hours_Required)

The table is in first normal form because it has no repeating attributes. The table is in second normal form because it has a single attribute key. If you assume Major → Hours_Required them Major is a determinate but not the key therefore you have a transitive dependency. To put the table in second normal form use the following tables.

STUDENT (SID, Stu_Name, P_O_Box, Major)

DEPARTMENT (Major, Hours_Required)

**Q.36. What are the special cases covered by BCNF but not by 3NF?**

BCNF covers two special cases not covered by 3NF: (1) part of a key determines part of a key and (2) a nonkey column determines part of a key.

**Q.37. The special cases covered by BCNF but not by 3NF are significant?**

The special cases are not significant because they rarely occur.

**Q.38. Is a relation with no non-key attribute is always in third normal form?**

This is a special case of 3NF. A 3NF relation typically involves a functional dependency between two non-key attributes, and if no such non-key attributes are present then the relation must be 3NF.

**Q.39. Can foreign keys contain null values? Explain why via an example.**

Yes foreign keys should be allowed to have null values because there are situations in which such kind of information is not available as follows:

- Employees that do not belong to a specific department, yet;
- Products that have not been put on the shelves, yet;
- Students that have not chosen some options, yet;

**Q.40. Define BCNF. Give an example of a relation in 3NF but not in BCNF. Transform the relation into relations in BCNF.**

A relation is in BCNF if every determinant is a candidate key. Consider this relation:

FAC-OFFICE (FID, Department, Building, Office).

Assume faculty members in the same department have offices in same building. The key is FID that determines Department, Building and Office. Department is not a candidate key and it determines Building. These relations are in BCNF:

FACULTY (FID, Department, Office)

DEPARTMENT-LOCATIONS (Department, Building)

**Q.41. Define fourth normal form. Give an example of a relation in BCNF but not in 4NF. Transform the relation into relations in 4NF.**

A relation is in 4NF if every determinant is a candidate key and it has no multi-valued dependencies. The following relation is in BCNF but not 4NF:

EMPLOYEE-HISTORY (Name, Project, PublicServiceActivity).

Project can be multi-valued because an employee could have worked on many projects. PublicServiceActivity can also be multi-valued. But Project and PublicServiceActivity are unrelated. These relations are in 4NF:

EMPLOYEE-WORK-HIST (Name, Project).

EMPLOYEE-SERVICE-HIST (Name, PublicServiceActivity)

**Q.42. What is the relationship between BCNF and 3NF? Is BCNF a stricter normal form than 3NF? Briefly explain your answer.**

Ans: BCNF is the revised 3NF definition. Yes BCNF is stricter than 3NF in that every table in BCNF is in 3NF but not every table in 3NF is in BCNF.

**Q.43. Why is the BCNF definition preferred to the 3NF definition?**

Ans: BCNF is the preferred definition because it is a simpler definition and provides the basis for the simple synthesis algorithm.

**Q.44. How many columns does an MVD involve?**

An MVD involves three columns.

**Q.45. What is a multivalued dependency (MVD)?**

A multivalued dependency is a relationship that can be derived from other relationships.

**Q.46. What is the relationship between MVDs and FDs?**

MVDs are generalizations of FDs. Every FD is an MVD, but not every MVD is an FD.

**Q.47. What is minimal normal form a relation must satisfy? Define this normal form.**

The minimal normal form is 1NF. A relation in which the intersection of each row and column contains one and only one value.

**Q.48. What is meant by 'union compatible'? Which operations require tables to be union compatible?**

The tables are 'union compatible' if they have the same degree (number of attributes) and the domain of the corresponding attributes is same. Union, intersection and difference operations require tables to be union compatible.

**Q.49. Why is it important to study the operators of relational algebra?**

Knowledge of operators can improve query formulation skills. The operators are fundamental ways to retrieve data from a relational database. The study of operators of relational algebra is important to become proficient in query formulation.

**Q.50. Why are the restrict and project operators widely used?**

These operators are widely used because users often want to see a subset rather than an entire table. These operators are also popular because they are easy to understand.

**Q.51. Why is the join operator so important for retrieving useful information?**

It is important for retrieving useful information because information resides in different tables and combining tables is important. The join operator is used to combine tables to retrieve information.

**Q.52. What happens to unmatched rows with the join operator?**

Unmatched rows are removed from the result of the join operator. Unmatched rows are included in the result of the full outer join operator.

**Q.53. State the difference between the following:**

a. Union & Intersection
b. Product & join
c. Selection and projection

a. The **Union** operation on two tables returns all rows from both the tables, but it does not repeat the duplicate rows. The Intersection operation on two table returns rows that exist in both tables involved.

b. A **product** of two table matches every row from the first table to each and every row in the second table. The **join** operation on two tables joins rows from two tables based on a common attribute, and the rows with same common attribute value are joined to each other.

c. The **Selection** operation on a table returns rows that satisfy the supplied criteria. The **projection** operation returns the attributes specified from a table.

**Q.54. What is the closure property of the relational algebra?**

The operators of the relational algebra are closed over relations, in the same sense as that in which those of arithmetic are closed over numbers. In arithmetic, plus, minus, times and divide each operates on a pair of numbers and when invoked yields a number. It is this so-called closure property that allows us to use an invocation of an operator as an operand of another invocation. By repeatedly doing that we can build numerical expressions of arbitrary complexity. In relational algebra, each operator operates on one or more relations and when invoked yields a relation, thus allowing relational expressions of arbitrary complexity to be written.

**Q.55. An organization works on a number of Projects, each identified by ProjectNo. There are number of employees identified by SIN who work on these projects. An employee can be working on more than one project at one time. The number of hours worked are also recorded. A preliminary table has been constructed as follows**

| SIN | ProjectNo | EmployeeName | ProjectName | EmpHours |
|---|---|---|---|---|

Assuming the relation to be in 1NF,

a. What should be the Primary Key? Is it Composite?
b. Write down the Partial dependencies.

c. Split the relation such that the resulting relations are in 3NF.

a) **(SIN, ProjectNo)**

b) **SIN → EmpoyeeName, ProjectNo → ProjectName**

c)

| SIN | ProjectNo | EmpHours |
|---|---|---|

| SIN | EmployeeName |
|---|---|

| ProjectNo | ProjectName |
|---|---|

**Q.56. A table contains sample data for parts and for vendors who supply parts. The part numbers uniquely identify parts and that vendor names uniquely identify vendors.**

| Part_No. | Description | Vendor_Name | Address | Unit_Cost |
|---|---|---|---|---|
| 1234 | Logic chip | Ejaz | Peoples colony | 10.00 |
| | | Naeem | Medina town | 8.00 |
| 5678 | Memory chip | Ali Raza | Peoples colony | 3.00 |
| | | Anjum | Raza Abad | 2.00 |
| | | Nasir | Saeed colony | 5.00 |

a. Convert this table to a relation (named PART SUPPLIER) in first normal form. Illustrate the relation with the sample data in the table.

b. List the functional dependencies in PART SUPPLIER and identify a candidate key.

c. For the relation PART SUPPLIER, identify each of the following: an insert anomaly a delete anomaly and a modification anomaly.

d. Draw a relational schema for PART SUPPLIER and show the functional dependencies.

e. In what normal form is this relation?

a. PART SUPPLIER

| Part_No | Description | Vendor_Name | Address | Unit_Cost |
|---|---|---|---|---|
| 1234 | Logic Chip | Ejaz | Peoples colony | 10.00 |
| 1234 | Logic Chip | Naeem | Medina town | 8.00 |
| 5678 | Memory Chip | Ali Raza | Peoples colony | 3.00 |
| 5678 | Memory Chip | Anjum | Raza Abad | 2.00 |
| 5678 | Memory Chip | Nasir | Medina town | 5.00 |

b.

Part_No → Description

Vendor_Name → Address

Part_No, Vendor_Name → Unit_Cost

c.

**Insert anomaly:** It is not possible to insert a new vendor before including a part number.

**Delete anomaly:** If part information is deleted, information about a vendor who supplies that part is also lost.

**Modification anomaly:** If a vendor address changes, all records for that vendor has to be modified.

d.


e. 1NF

**Q.57. Examine the Patient Medication Form for Civil Hospital as follows:**

**Civil Hospital**
**Patient Medication Form**

**Patient Number: 9876**

**Full Name: Ali Ahmad** **WardNumber: W11**
**BedNumber: 87** **Ward Name: Fatima**

| Drug Number | Name | Description | Dosage | Method of Admin | Units per Day | Start Date | Finish Date |
|---|---|---|---|---|---|---|---|
| 10223 | Morphine | Pain killer | 10mg/ml | Oral | 50 | 24/03/01 | 25/04/02 |
| 10334 | Tetracycline | Antibiotic | 0.5mg/ml | IV | 10 | 24/03/01 | 17/04/01 |
| 10223 | Morphine | Pain killer | 10mg/ml | Oral | 10 | 25/04/02 | 02/05/03 |

a. Identify the functional dependencies represented by the data shown in the form in
b. Describe and illustrate the process of normalizing the data shown in Figure to first (1NF), second (2NF), third (3NF), and BCNF
c. Identify the primary, alternate, and foreign keys in your BCNF relations

**a)**

Patient No → Full Name

Ward No → Ward Name

Drug No → Name, Description, Dosage, Method of Admin

Patient No, Drug No, Start Date → Units per Day, Finish date

Functional dependencies for Bed No are unclear. If Bed No is unique number for entire hospital then Bed No → Ward No. Bed No is concerned with allocation of patients on waiting list to beds.

**b)**

**First Normal Form**

Patient No, Drug No, Start Date, Full Name, Ward No, Ward Name, Bed No, Name, Description, Dosage, Method of Admin, Units per Day, Finish Date

**Second Normal Form**

Patient No, Drug No, Start Date, Ward No, Ward Name, Bed No, Units per Day, Finish Date

Drug No, Name, Description, Dosage, Method of Admin

Patient No, Full Name

**Third Normal Form/BCNF**

Patient No, Drug No, Start Date, Ward No, Bed No, Units per Day, Finish Date

Drug No, Name, Description, Dosage, Method of Admin

Patient No, Full Name

Ward No, Ward Name

c)

Patient No(FK), Drug No(FK), Start Date, Ward No(FK), Bed No, Units per Day, Finish Date

Drug No, Name, Description, Dosage, Method of Admin

Patient No, Full Name

WARD (Ward No, Ward Name)

The primary keys underlined.

**Q.58. Consider the following relation definition and sample data:**

| ProjectID | EmployeeName | EmployeeSalary |
|---|---|---|
| 100a | Adnan | 65000 |
| 100a | Salman | 5100 |
| 100b | Salman | 51000 |
| 200a | Adnan | 64000 |
| 200b | adnan | 64000 |
| 200c | Amir | 28000 |
| 200c | Salman | 51000 |
| 200d | Amir | 28000 |

PROJECT (ProjectID, EmployeeName, EmployeeSalary)

where ProjectID is the name of a work project

EmployeeName is the name of an employee who works on that project

EmployeeSalary is the salary of the employee whose name is EmployeeName

Assuming that all of the functional dependencies and constraints are apparent in data, which of the following statements is true?

A. **ProjectID → EmployeeName**
FALSE: There are multiple Employee Names for each project. (see project 100A)

B. **ProjectID → EmployeeSalary**
FALSE: There are multiple Employee Salaries for each project. (see project 100a)

C. **(ProjectID, EmployeeName) → EmployeeSalary**
FALSE: Each employee's salary is always the same, regardless of the project. (see salman)

D. **EmployeeName → EmployeeSalary**
TRUE: An Employee Name always has the same Salary. (see salman)

E. **EmployeeSalary → ProjectID**
FALSE: There are multiple ProjectIDs for a given Salary. (see Salary 51000)

F. **EmployeeSalary → (ProjectID, EmployeeName)**
FALSE: There are multiple ProjectID and EmployeeName combinations for a given Salary (see Salary 51000)

**Answer these questions:**

G. **What is the key of PROJECT?**

ProjectID and EmployeeName (Composite Key)

H. **Are all non-key attributes (if any) dependent on the whole key?**
No: EmployeeSalary is non-key attribute and it is dependent on EmployeeName only.

I. **In what normal form is PROJECT?**

Project is in first normal form because it has no multi-valued attributes. It is not in second normal form because it has a partial dependency. Key: ProjectID+EmployeeName but EmployeeName → EmployeeSalary.

J. **Describe two modification anomalies from which PROJECT suffers.**
**Insertion Anomaly:** It is not possible to add an Employee until Employee is assigned to a Project. Likewise, you cannot add a Project until and Employee is assigned to the Project.

**Update Anomaly:** If you want to change Salman's Salary you need to change three rows of data in order to change one Employee's salary.

**Deletion Anomaly:** If Amir did not work on Project 200C and worked in Project 200D only, deletion of ProjectC would delete the fact that Amir's salary was 28000.

K. **Is ProjectID a determinant?**

No

L. **Is EmployeeName a determinant?**

YES: EmployeeName → EmployeeSalary

M. **Is (ProjectID, EmployeeName) a determinant?**

No

N. **Is EmployeeSalary a determinant?**

No: In this case, it appears that it can be a determinate because no two people have same salary. Using logic however, one can assume that there is no business rule in a firm that says two people cannot have the same salary.

O. **Does this relation contain a partial dependency? If so, what is it?**

YES, the relation does contain a partial dependency. The key is ProjectID+EmployeeName but EmployeeName → EmployeeSalary.

P. **Redesign this relation to eliminate the modification anomalies.**

PROJECT (ProjectID, EmployeeName)
EMPLOYEE (EmployeeName, EmployeeSalary)

**Q.59. Consider the following relation definition and sample data:**

**PROJECT-HOURS Relation**

| EmployeeName | ProjectId | TaskID | Phone | Total Hours |
|---|---|---|---|---|
| SALMAN | 100A | B-1 | 788792 | 12 |
| SALMAN | 100A | P-1 | 788792 | 12 |
| SALMAN | 200B | B-1 | 788792 | 12 |
| SALMAN | 200B | P-1 | 788792 | 12 |
| ADNAN | 100A | C-1 | 788988 | 26 |
| ADNAN | 200A | C-1 | 788988 | 26 |
| ADNAN | 200D | C-1 | 788988 | 26 |

PROJECT-HOURS (EmployeeName, ProjectID, TaskID, Phone, TotalHours)
where EmployeeName is the name of an employee
ProjectID is the name of a project
TaskID is the name standard work task
Phone is the employee's telephone number
TotalHours is the hours worked by the employee on this project
Assuming that all of the functional dependencies and constraints are apparent in this data, which of the following statements is true?

A. **EmployeeName → ProjectID**

NO: There are multiple ProjectIDs for each EmployeeName

B. **EmployeeName →→ ProjectID**

YES: Each EmployeeName has three or more ProjectIDs

C. **EmployeeName → TaskID**

NO: There are multiple TaskIDs for each EmployeeName

D. **EmployeeName →→ TaskID**

YES: Salman has multiple (2) TaskIDs

E. **EmployeeName →: Phone**

YES: Each EmployeeName has exactly one Phone value

F. **EmployeeName →: TotalHours**

YES: Each EmployeeName has exactly one TotalHours value

**G. (EmployeeName, ProjectID) → TotalHours**

YES: It is true based on fourth assumption. Looking at data only, it is more probable that EmployeeName → TotalHours. It is because TotalHours is same for EmployeeName regardless of ProjectID.

**H. (EmployeeName, Phone ) → TaskID**

NO: There are multiple TaskIDs for a given EmployeeName, Phone combination

**I. ProjectID → TaskID**

NO: There are multiple TaskIDs for a given ProjectID

**J. TaskID → ProjectID**

**NO: There are multiple ProjectIDs for a given TaskID**

**Answer these questions:**

**K. What are all of the determinants?**

EmployeeName → Phone
EmployeeName → → TaskID
EmployeeName, ProjectID → TotalHours

**L. Does this relation contain a partial dependency? If so, what is it?**

YES, it contains a partial dependency. The key is EmployeeName + ProjectID + TaskID but EmployeeName → Phone. So there is a partial dependency.

**M. Does this relation contain a multi-value dependency? If so, what are the unrelated attributes?**

YES: The related attributes are ProjectID and TaskID. EmployeeName → → ProjectID and EmployeeName → → TaskID

**N. What is the deletion anomaly that this relation contains?**

If Employee Salman no longer has TaskID B-1 two rows must be deleted, Row 1 and Row 3. The deletion of one fact requires deletion or two rows.

**O. How many themes does this relation have?**

It would appear that there are at least three themes. 1) Employees and their Tasks 2) Employees and their Phone Numbers and 3) Employees and hours worked on a project.

**P. Redesign this relation to eliminate the modification anomalies. How many relations did you use? How many themes does each of your new relations contain?**

EMPLOYEE (EmployeeName, Phone)
EMPLOYEE_TASKS (EmployeeName, TaskID)
PROJECT-HOURS (EmployeeName, ProjectID, TotalHours)

Three relations are required, one for each theme. Each relation now carries one theme

---

## Multiple Choice

1. **The columns of a table correspond to:**
   a. Table b. Record c. Field d. Cell
2. **The foreign key is found in:**
   a. Parent table b. Dependent table c. Pivot table d. Index table
3. **A table must have:**
   a. Primary key b. secondary key c. Composite key d. Sort key
4. **A two-dimensional table of data is called a:**
   A. Group b. Set c. Declaration d. Relation
5. **A table is perceived as two-dimensional structure composed of:**
   a. x and y co-ordinates b. Matrix elements
   c. Rows and columns d. Intersection of data
6. **A relation is analogous to a:**
   a. File b. Field c. Record d. Row

7. **A relation is also known as:**
   a. Table b. Tuple c. Relationship d. Attribute
8. **An attribute is also known as a:**
   a. Table b. Relation c. Row d. Field
9. **A row of a relation is called a (n):**
   a. Attribute b. Entity c. Tuple d. Both a and c
10. **A person, place, thing, event, or condition about which data is kept in the database is called:**
    a. Attribute b. Field c. Record d. Entity
11. **A category of data or information that describes an entity is called a(n):**
    a. Attribute b. Data item c. Record. d. Tuple
12. **Which of the following is NOT a general characteristic of relations?**
    a. Each row is unique. b. The order of columns is significant
    c. The order of rows is insignificant d. Columns are all elemental or atomic.
13. **An index can be used to:**
    a. Improve the performance of the database
    b. Document the structure of the database itself
    c. Reduce data dependency for application programs
    d. All
14. **Which of the following are properties of relations?**
    a. Each attribute has a unique name
    b. No two rows in a relation are identical
    c. There are no multivalued attributes in a relation
    d. All of the above
15. **A key is:**
    a. a field that identifies only one record b. the most important field in a record
    c. the first field of table d. None
16. **Which of the following describes the primary key?**
    a. It must be unique b. It helps in indexing of a large database
    c. It makes sorting quicker d. All of the above
17. **Which of the following is NOT a good primary key?**
    a. Social security number b. Order number c. Zip code d. Student ID number
18. **How many primary keys can a table have?**
    a. One b. At least one, but not more than two c. Between 1 and 5 d. No limit
19. **Which of the following should be a primary key?**
    a. A person's last name b. An employee's salary
    c. A customer's ID number d. A salesperson's region
20. **Which field listed below is the most appropriate primary key?**
    a. A person's name b. A person's street address
    c. A person's birth date d. A person's Social Security Number
21. **One field or combination of fields for which more than one record may have the same combination of values is called:**
    a. Secondary key b. Index c. Composite key d. Linked key
22. **A candidate key is:**
    a. Primary key
    b. The primary key selected to be the key of a relation
    c. An attribute or group of attributes that can be the primary key
    d. All

**23. An attribute in a relation of a database that serves as the primary key of another relation in the same database is called a:**
a. Global key b. Link key c. Foreign key d. None

**24. A primary key that consists of more than one attribute is called a:**
a. Foreign key b. Composite key c. Multivalued key d. Global key

**25. In 3NF, which form of dependency is removed?**
a. Functional b. Non-functional c. Associative d. Transitive

**26. In relational database, table is also called:**
a. Tuple b. Relation c. File d. Schema

**27. In 3NF, a non-key attribute must not depend on a:**
a. Non-key attribute b. key attribute c. Composite key d. Sort key

**28. Different attributes in two different tables having same name are referred to as:**
a. Synonym b. Homonym c. Acronym d. Mutually exclusive

**29. Every relation must have:**
a. Primary key b. Candidate key c. Secondary key d. Mutually exclusiveness

**30. The entity integrity rule states that:**
a. No primary key attribute can be null b. Each entity must have a primary key
c. Primary key must have only one attribute. d. None

**31. A rule that states that each foreign key value must match a primary key value in the other relation is called:**
a. Referential integrity constraint b. Key match rule
c. Entity key group rule d. Foreign/primary match rule

**32. Two or more attributes having different names but same meaning are called:**
a. Homonyms. b. Aliases c. Synonyms d. Alternate attributes

**33. A constraint between two attributes is called a(n):**
a. Functional relation b. Attribute dependency.
c. Functional dependency. d. Functional relation constraint.

**34. The attribute on the left-hand side of the arrow in a functional dependency is:**
a. Candidate key b. Determinant c. Foreign key d. Primary key

**35. The goal of normalization is to:**
a. Get stable data structure b. Increase number of relation c. Increase redundancy d. None

**36. A relation is in 2NF if it is in 1NF and all its non-key attributes are:**
a. Dependent on part of the primary key b. Dependent on the entire primary key
c. Independent of the primary key d. Independent of any other relation

**37. In 2NF, which form of dependency is removed?**
a. Functional b. Partial c. Associative d. Transitive

**38. Which of the following are anomalies that can be caused by redundancy in tables?**
a. Insertion b. Deletion c. Modification d. All

**39. A functional dependency between two or more non-key attributes is called:**
a. Partial functional dependency b. Partial non-key dependency
c. Transitive dependency d. None

**40. Which of the following anomalies result from a transitive dependency?**
a. Insertion b. Modification c. Deletion d. All

**41. A relation is in third normal form if it is in second normal form and:**
a. Dependent on part of the key b. Dependent on all of the key
c. Independent of the key d. Has no transitive dependencies

**42. A relation that contains minimal redundancy and allows easy use is called:**
a. Clean. b. Simple. c. Complex. d. Well-structured.

**43. The 1NF describes the tabular format in which:**
a. All the key attributes are defined
b. No repeating groups in the table
c. All attributes are dependent on the primary key
d. All

**44. In a functional dependency, the determinant:**
a. Will be paired with one value of the dependent attribute
b. May be paired with one or more values of the dependent attribute
c. May consist of more than one attribute
d. a and c

**45. When the determinant contains two attributes:**
a. The first attribute determines the dependent attribute
b. The second attribute determines the dependent attribute
c. Both attributes determine the dependent attribute
d. Either the first or second attribute determines the dependent attribute

**46. An anomaly in a relation is:**
a. An unusual data value
b. A duplicate data value caused by changing the data
c. An undesirable consequence of changing the data
d. An error in the design

**47. Restrictions on operations on a relation are called:**
a. Deletion anomalies
b. Insertion anomalies
c. Modification anomalies
d. Referential integrity constraints

**48. The normalization process generally:**
a. Reduces the number of relations
b. Increases the number of relations
c. Reduces the number of functional dependencies
d. Increases the number of functional dependencies

**49. A relation is automatically in:**
a. First Normal Form
b. Second Normal Form
c. Third Normal Form
d. Boyce-Codd Normal Form

**50. A relation is in domain/key normal form if:**
a. Every key of relation is a logical consequence of definition of constraints and determinants
b. Every key of relation is a logical consequence of definition of constraints and domains
c. Every constraint on relation is a logical consequence of definition of keys and determinants
d. Every constraint on relation is a logical consequence of definition of keys and domains

**51. Which of the following is TRUE from functional dependency shown as (A, B) → (C, D)?**
a. A is the determinant of C
b. A and B together are determined by C and D together
c. A and B together determine D
d. C and D together determine A

**52. What is the highest normal form a relation is in if every determinant is a candidate key?**
a. First
b. Second
c. Third
d. BCNF

**53. A relation is considered a:**
a. Column
b. One-dimensional table.
c. Two-dimensional table
d. Three-dimensional table

**54. A functional dependency is a relationship between or among**
a. Tables
b. Rows
c. Columns
d. Attributes

**55. Which of the following is a group of one or more attributes that uniquely identifies a row?**
a. Key
b. Determinant
c. Tuple
d. Relation

**56. For some relations, changing the data can have undesirable consequences, called:**
a. Referential integrity constraints. b. Modification anomalies
c. Normal forms d. Normal forms

**57. When the values in one attribute must exist in another attribute, it is called:**
a Transitive dependency b. Insertion anomaly
c. Referential integrity constraints d. Normal forms

**58. The different classes of relations and the techniques for preventing anomalies are called:**
a. Normal forms b. Referential integrity constraints
c. Modification anomalies d. None of the above

**59. Two or more attributes or attribute collections that can be a key are called**
a. Candidate keys b. Determinants c. Primary keys d. BCNF

**60. A relation is in this form if it is in BCNF and has no multi-value dependencies**
a. Second b. Third c. Fourth d. Domain/key normal form.

**61. If attribute A determines attribute B and B determines A, the values of the attributes have this relationship:**
a. One-to-one b. Many-to-one c. Normalized d. Many-to-many

**62. If attribute A determines B but B does not determine A, the relationship among their data values is:**
a. One-to-one relationship b. Many-to-one relationship
c. Normalized relationship d. Many-to-many relationship

**63. One solution to the multi-value dependency constraint problem is to:**
a. Split relation in two relations, each with a single theme b. Change the theme
c. Create a new relation d. Add a composite key

**64. In ___ normal form, any multi-valued attributes have been removed:**
a. First b Second c. Fourth d. Fifth

**65. A relation is in fourth normal form if it is in BCNF and it has no:**
a. Deletion dependencies. b. Transitive dependencies.
c. Multi-value dependencies d. Partial dependencies.

**66. A partial functional dependency (FD) means that:**
a. Some attributes of an entity are not known
b. Not all attributes on right-hand side of FD are necessary
c. No dependency exists in the entity
d. Not all of the attributes on the left-hand side of the FD are necessary

**67. The anomalies addressed by moving from BCNF to 4NF generally deal with:**
a. Excessive updates and redundancy of data for each entity
b. Inability to uniquely identify an entity
c. Inability to reconstruct relations once they have been decomposed.
d. Creation of identical rows in a relation

**68. In general, a row in a relation should have all of the data about**
a. One instance of the relation's theme b. Each instance of the relation's theme
c. Every instance of the relation's theme d. No instance of the relation's theme

**69. Which of the following is a requirement of 3NF?**
a. Must contain a partial dependency. b. Must contain a composite.
c. Must contain no transitive dependencies Must contain no partial dependencies

**70. Which is TRUE from functional dependency shown as A → (X, Y)?**
a. X is functionally dependent on A b. A determines Y
c. A is a determinant d. X and Y are functionally dependent on A
e. All of the above

**71. Which of the following should not be placed in a relational table?**
a. Entity b. Attribute c. Relationship d. Repeating group

**72. In BCNF, every ____ in a table is a candidate key.**
a. determinant b. entity c. primary key d. atomic attribute

**73. An index:**
a. Makes retrievals faster
b. Is always generated for a primary key
c. Increases the space needed for the database
d. Both a and b

**74. A 3NF violation means:**
a. An attribute in the table can have several values in one record
b. An attribute in the table depends on only one part of a concatenated key
c. An attribute in the table belongs in another entity
d. The table has no primary key

**75. Normalization:**
a. Identifies incorrect referential integrity
b. Identifies incorrect placement of attributes
c. Identifies incorrect cardinality
d. Identifies incorrect foreign keys

**76. Normalized databases**
a. Are designed for efficient update
b. Remove key/foreign key redundancy
c. Remove data redundancy
d. Both a and c

**77. A relation is not allowed to have:**
a. Two columns with the same values
b. Two columns with the same domain
c. Two columns with the same names
d. None of these

## Answers

| | | | | | |
|---|---|---|---|---|---|
| 1. c | 2. b | 3. a | 4. d | 5. c | 6. a |
| 7. a | 8. d | 9. c | 10. d | 11. a | 12. b |
| 13. a | 14. d | 15. a | 16. d | 17. c | 18. a |
| 19. c | 20. d | 21. a | 22. c | 23. c | 24. b |
| 25. d | 26. b | 27. a | 28. b | 29. a | 30. a |
| 31. a | 32. c | 33. c | 34. b | 35. a | 36. b |
| 37. b | 38. d | 39. c | 40. d | 41. d | 42. d |
| 43. d | 44. d | 45. c | 46. c | 47. d | 48. b |
| 49. a | 50. d | 51. c | 52. d | 53. c | 54. d |
| 55. a | 56. b | 57. c | 58. a | 59. a | 60. c |
| 61. a | 62. b | 63. a | 64. a | 65. c | 66. d |
| 67. a | 68. a | 69. c | 70. e | 71. d | 72. a |
| 73. d | 74. c | 75. b | 76. d | 77. c | |

## True / False

1. A relation is a three-dimensional table
2. A characteristic of a relation is that the cells of the relation hold a single value.
3. A characteristic of a relation is that the rows of a relation may hold identical values
4. The columns of a relation are sometimes called "tuples."
5. Keys are always unique
6. A tuple is a group of one or more columns that uniquely identifies a row
7. A row can be uniquely identified by a key
8. A key can be composed of a group of attributes taken together
9. It is possible to have a relation that does not have a key
10. Attribute Y is functionally dependent on attribute X if value of X determines value of Y
11. The functional dependency A → B means that value of A can be determined from value of B.
12. In the functional dependency shown as A → B, B is the determinant
13. Functional dependencies can involve groups of attributes
14. A determinant of a functional dependency may or may not be unique in a relation
15. Normalization is the process of splitting a relation into two or more relations.
16. A functional dependency is a relationship between or among attributes.
17. The known or given attribute is called the determinant in a functional dependency.
18. The relationship in a functional dependency is one-to-one (1:1).
19. The selection of the attributes to use for the key is determined by the database programmers.
20. A deletion anomaly occurs when deleting one entity results in deleting facts about another entity.
21. An insertion anomaly occurs when we cannot insert some data into the database without inserting another entity first.
22. Modification anomalies do not occur in tables that meet the definition of a relation.
23. A table of data that meets the minimum definition of a relation is automatically in first normal form.
24. A relation is in first normal form if all of its non-key attributes are dependent on part of key.
25. A relation is in second normal form if all of its non-key attributes are dependent on entire key.
26. A relation can be in third normal form without being in second normal form.
27. Fifth normal form is the highest normal form.
28. Normalization decomposes relations to produce small, well-structured relations
29. A determinant is a constraint between two attributes or sets of attributes
30. A determinant is an attribute or set of attributes that uniquely identifies a row in a relation.
31. A determinant of a functional dependency will always be unique in a relation.
32. A functional dependency is a relationship between or among attributes.
33. Every relation has at least one key
34. Classes of relations and the techniques for preventing anomalies are called normal forms..
35. A relation is in 4th normal form if it is in 2nd normal form and has no transitive dependencies.
36. A relation is in BCNF if every determinant is a candidate key.
37. A key is a unique identifier of an attribute
38. If two attributes functionally determine each other, the relationship of their data values is one to many.
39. If two attributes have a one-to-one relationship, they functionally determine each other
40. If A determines B and B does not determine A, the relationship among their values is many-to-many.
41. If attribute A determines B but B does not determine A, the relationship among their data values is many to one
42. Normalized relations avoid modification anomalies.
43. Anomalies can be removed by splitting the relation into two or more relations
44. When a key of one relation is stored in a second relation, it is called a foreign key
45. A null value is an attribute value that has never been supplied.

46. If A → B and B → A, then A and B have a many-to-many attribute relationship.
47. Fifth normal form deals with obscure problems with transitive dependencies
48. A relation is in 4NF if it is in 3NF and contains no multi-value dependencies.
49. A relation can have only one candidate key
50. A relation is in 3NF if it is in 2NF and contains no transitive dependencies
51. A transitive dependency exists when a non-key attribute is determined by only part of the key.
52. Relations that have a composite key and are in 1NF are automatically in 2NF.
53. Any table that meets the definition of a relation is in 2NF
54. Relations are classified into "normal forms" based on the types of modification anomalies that they are vulnerable to
55. Breaking a relation into two or more relations may create the need for a referential integrity constraint to be defined
56. If a table meets minimum definition of a relation, it has an effective or appropriate structure.
57. Functional dependencies can involve groups of attributes
58. It is possible to have a relation that does not have a key.
59. Indexes makes the searching of a record faster
60. One property of a relation is that each attribute within a relation has a unique name.
61. In 2NF, every non-key attribute must depend on the key attribute.
62. Partial dependencies are removed in 3NF.
63. The database is normalized to avoid certain database anomalies.
64. A database anomaly leads the database on to an inconsistent state.
65. Relational models view the data as part of a table or collection of tables in which all key values must be identified.
66. Normalization produces a lower normal form.

## Answers

| | | | |
|---|---|---|---|
| 1. T | 2. T | 3. F | 4. F |
| 5. F | 6. F | 7. T | 8. T |
| 9. F | 10. T | 11. F | 12. F |
| 13. T | 14. T | 15. T | 16. T |
| 17. T | 18. F | 19. F | 20. T |
| 21. T | 22. F | 23. T | 24. F |
| 25. T | 26. F | 27. F | 28. T |
| 29. F | 30. F | 31. F | 32. T |
| 33. T | 34. T | 35. F | 36. T |
| 37. F | 38. F | 39. T | 40. F |
| 41. T | 42. T | 43. T | 44. T |
| 45. T | 46. F | 47. F | 48. T |
| 49. F | 50. T | 51. F | 52. F |
| 53. F | 54. T | 55. T | 56. F |
| 57. T | 58. F | 59. T | 60. T |
| 61. T | 62. F | 63. T | 64. T |
| 65. T | 66. F | | |

CHAPTER 6

# Database Design using E–R Model

## Chapter Overview

# 6.1 Database Design using E-R Model

E-R model represents different things as **entities**. The connections among different entities are represented by **relationships**. These entities and relationships can be transformed into **relational model**. This model can be used to design the database using relational model

## 6.1.1 Converting Entities into Relations

The process of converting entities into relations is very simple. In this process, the name of entity becomes the name of relation and attributes of entity becomes the fields of relation. After this simple conversion, the relation is examined according to the normalization criteria. The relation can be modified if it does not satisfy the rules of normalization.

The following are some vocabulary that are commonly used. Note the different terms used depending on the model being discussed.

| ER Model | Relational Model | Database | Traditional Programmer |
|---|---|---|---|
| Entity | Relation | Table | File |
| Entity Instance | Tuple | Row | Record |
| Attribute | Attribute | Column | Field |
| Identifier | Key | Key | Key (or link) |

### Example

Following example explains the process of converting an entity into a relation:

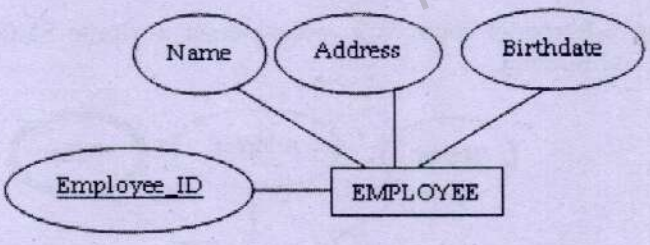



**Figure:** EMPLOYEE entity

| | | | |
|---|---|---|---|
| (EmployeeID | Name | Address | Birthdate) |

**Figure:** EMPLOYEE Relation

In the above example, EMPLOYEE entity is converted into relation. The attributes of the entity are fields of the relation. The data model describes EmployeeID as an identifier and is underlined. The above relation is using EmployeeID as primary key for the relation.

## 6.1.2 Converting Composite Attributes

If an entity contains composite attribute, each part of the attribute is represented by a separate field in the relation.

### Example

The following E-R model contains a composite attribute Address. Different parts of this attributes are used as fields in the relation.

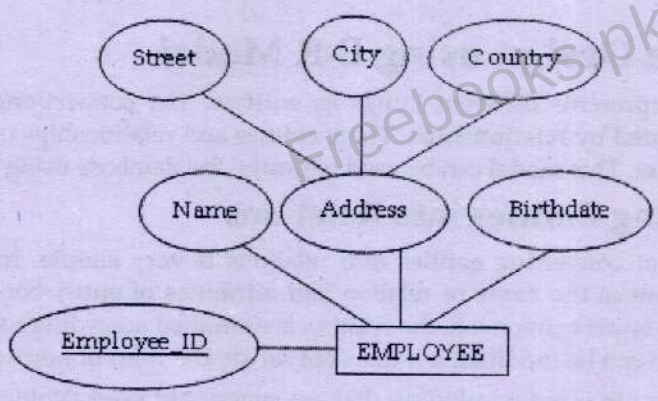



**Figure:** EMPLOYEE entity

| (EmployeeID | Name | Birhtdate | Street | City | Country |
|---|---|---|---|---|---|

**Figure:** EMPLOYEE relation

## 6.1.3 Converting Multi-valued Attributes

The attribute is represented in a separate relation if an entity contains multi-valued attribute.

### Example

The following E-R model contains a multi-valued attribute Skills. This attribute is represented in a second relation.

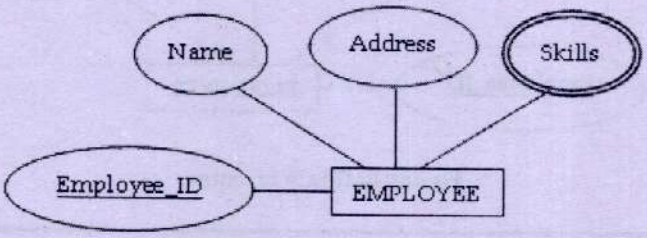



**Figure:** EMPLOYEE entity with multi-valued attribute Skills

| EmployeeID | Name | Birhtdate | Street | City | Country |
|---|---|---|---|---|---|

**Figure:** EMPLOYEE relation

| EmployeeID | Skill |
|---|---|

**Figure:** SKILLS relation

The second relation SKILLS is using EmployeeID and Skill as composite key. It may contain many rows. Each row will contain a single skill of the employee.

## 6.1.4 Converting Weak Entities

A weak entity does exist independently. It depends on the existence of another entity known as **identifying owner**. When entities are converted into relations, first of all a relation for the identifying owner is created. A separate relation is created for each weak entity and attributes of weak entity become the fields of the relation. The weak entity relation is connected with the identifying relation. The primary key of identifying relation is used as foreign key in the weak entity relation.

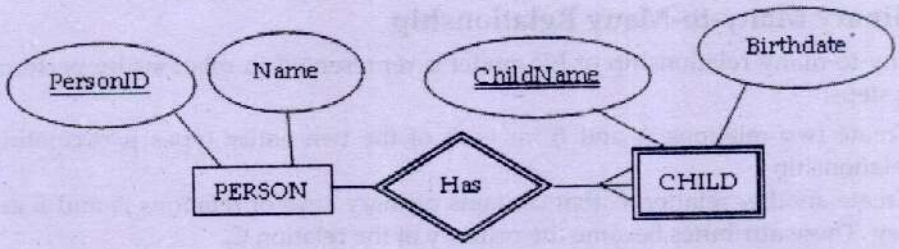



**Figure:** Weak entity CHILD

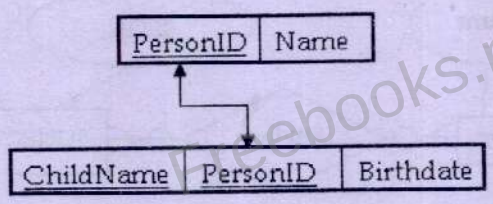



**Figure:** PERSON and CHILD relation

## 6.1.5 Converting Binary Relationship

The process of representing relationships depends upon two things:

1. Degree of relationship
2. Cardinality of relationship

### 6.1.5.1 Binary One-to-Many Relationship

One-to-many relationship of ER model is represented in relations by performing the following two steps:

1. Create a relation for each of the two entity types participating in the relationship
2. Include the primary key of the entity on one-side of the relationship as a foreign key in the relation that is on the many-side of the relationship.

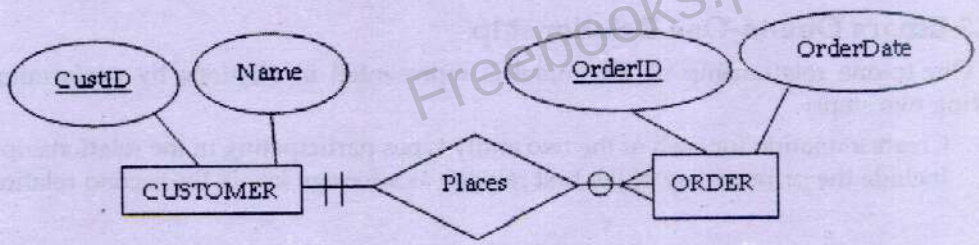



**Figure:** Binary One-to-Many Relationship

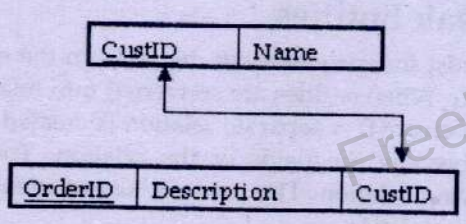



**Figure:** Relations in Binary One-to-Many Relationship

### 6.1.5.2 Binary Many-to-Many Relationship

Many-to-many relationship of ER model is represented in relations by performing the following steps:

1. Create two relations A and B for each of the two entity types participating in the relationship
2. Create another relation C that contains primary keys of relations A and B as foreign key. These attributes become the primary of the relation C.

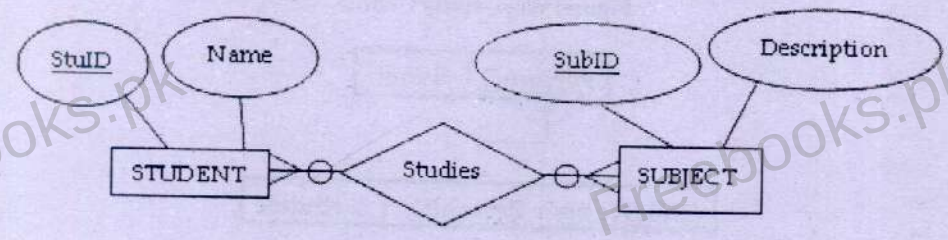



**Figure:** Binary Many-to-Many Relationship

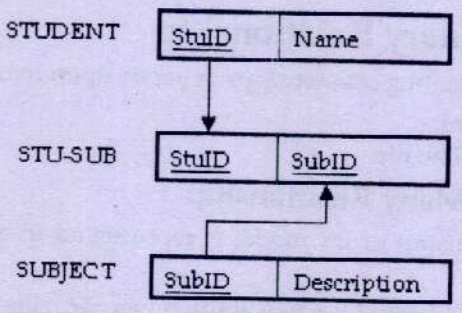



**Figure:** Relations in Binary Many-to-Many Relationship

### 6.1.5.3 Binary One-to-One Relationship

One-to-one relationship of ER model is represented in relations by performing the following two steps:

1. Create a relation for each of the two entity types participating in the relationship
2. Include the primary key of the first relation as a foreign key in the second relation.

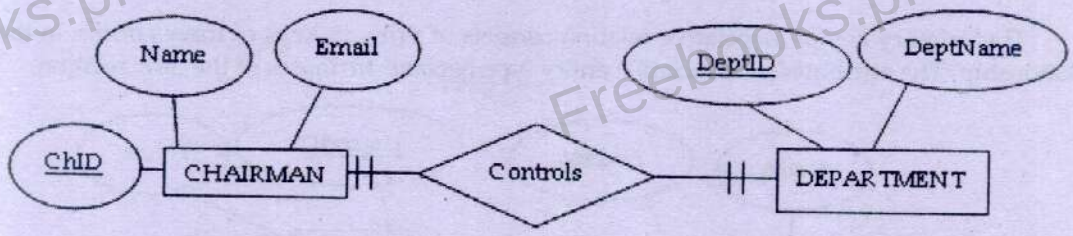



**Figure:** Binary One-to-One Relationship

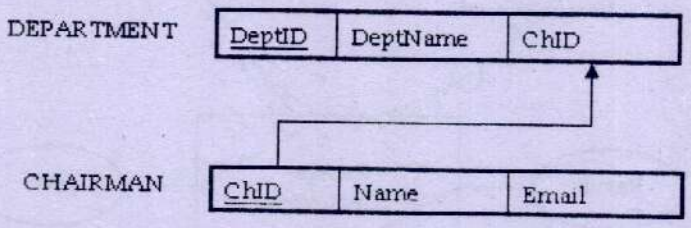



**Figure:** Relations in Binary one-to-one relationship

## 6.1.6 Unary Relationships

Unary relationship exists between the instances of same entity types. It is also known as **recursive relationship**. Unary one-to-many relationship of ER model is represented in relations by performing the following two steps:

1. Create a relation to represent the entity type.
2. Add another field as foreign key in the same relation that references the primary key of the relation. The foreign key must have the same domain as the primary key.

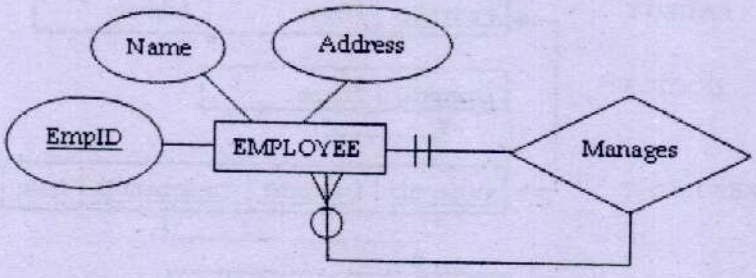



**Figure:** Unary one-to-many relationship

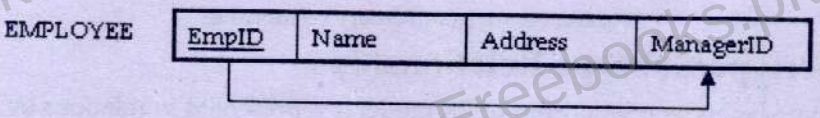



**Figure:** Relation in unary one-to-many relationship

## 6.1.7 Ternary Relationships

Ternary relationship exists between the instances of three entity types. Ternary relationship of ER model is represented in relations by performing the following two steps:

1. Create a relation for each entity type participating in the relationship.
2. Create an associative relation to represent the link between three entities.

The primary key of associative relation consists of primary keys of three entities in the relationship. The attributes of associative entity type become attributes of the new relation.

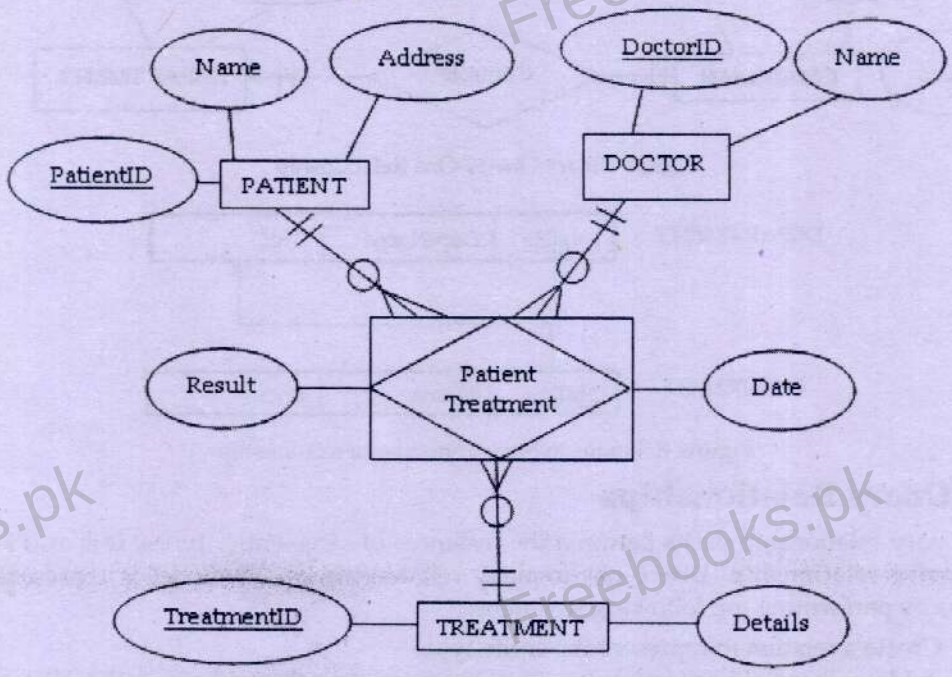



**Figure:** Ternary relationship

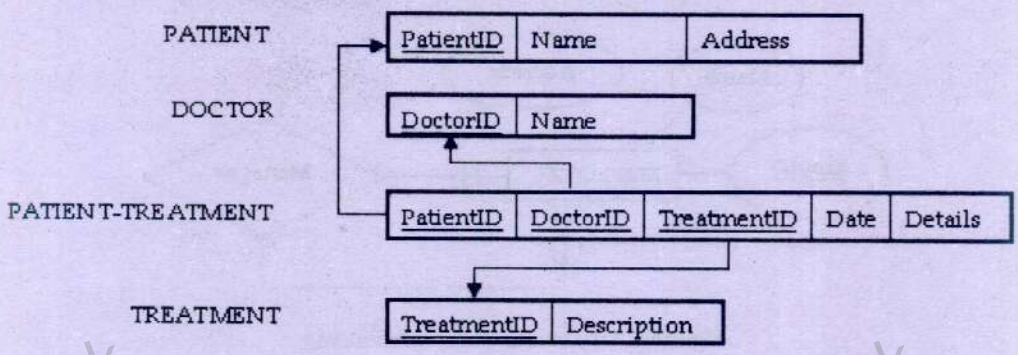



**Figure:** Relations in Ternary relationship

## 6.1.8 Supertype/Subtype Relationships

Supertype/subtype relationship of ER model is represented in relations by performing the following steps:

1. Create a separate relation for supertype and for each of subtypes.
2. The relation for supertype consists of the attributes which are common in all members. Assign a primary key to the supertype relation.
3. The relation for each subtype consists of the attributes, which are unique to that particular subtype. The primary key of the subtype will be similar to the primary key of the supertype.
4. Assign one or more attributes of the supertype to work as the subtype discriminator.

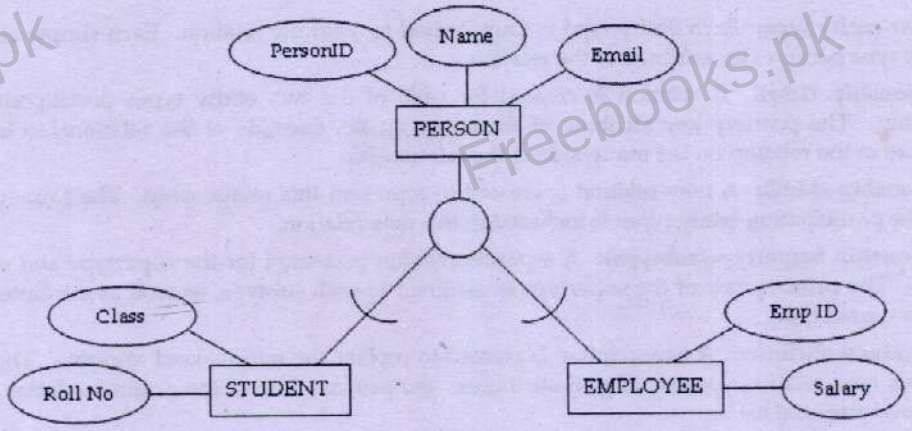



**Figure:** Supertype/Subtype relationship

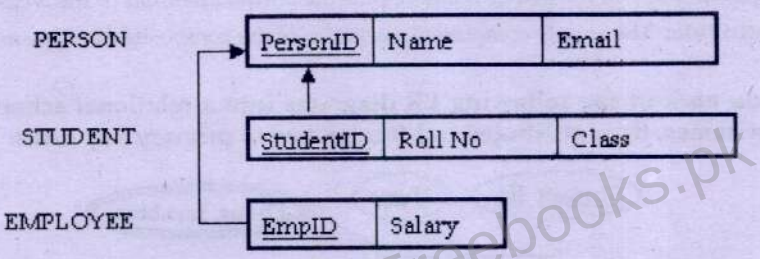



**Figure:** Supertype/Subtype relationship

## Short Questions

**Q.1. How is the relationship between entities represented in the relation data Model.**

Relationships between entities are represented by foreign key values in one relation that match primary key values in another relation.

**Q2. How do you represent a 1:M unary relationship in a relation data Model?**

A 1:M unary relationship is represented by a recursive foreign key whose values reference the primary key values of the same relation.

**Q.3. How do you represent a M: N unary relationship in a relation data Model**

An M:N ternary relationship is represented by a new associative relation whose primary key consists of the primary key attributes of the participating entity types.

**Q.4. Describe how the following components of an E-R Diagram are transferred to relations**

a) Regular entity type
b) Relationship (1:M)
c) Relationship (M:N)
d) Relationship (supertype/subtype)
e) Multivalue attribute
f) Weak Entity
g) Composite Attribute

**a. Regular entity type:** Each entity type is transformed to a simple relation. Each simple attribute of the entity type becomes an attribute of the relation.

**b. Relationship (1:M):** A relation is created for each of the two entity types participating in the relationship. The primary key attribute of the entity on the one-side of the relationship becomes a foreign key in the relation on the many-side of the relationship.

**c. Relationship (M:N):** A new relation is created to represent this relationship. The primary key for each of the participating entity types is included in this new relation.

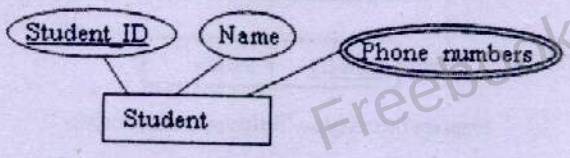
**d. Relationship (supertype/subtype):** A separate relation is created for the supertype and each of its subtypes. The primary key of the supertype is assigned to each subtype, as well as attributes that are unique to the subtype.

**e. Multivalued attribute:** A new relation is created to replace the multivalued attribute. The primary key of this new relation consists of two attributes: the primary key of the original relation, plus the multivalued attribute itself.

**f. Weak entity:** A new relation is created corresponding to weak entity. The primary key of this relation consists of the primary key of the owner relation, plus the partial identifier of the weak entity type.

**e. Composite attribute:** The simple component attributes of the composite attribute are included in the new relation

**Q.5. Transform each of the following ER diagrams into a relational schema showing the relation names, their attributes and foreign key to primary key links:**



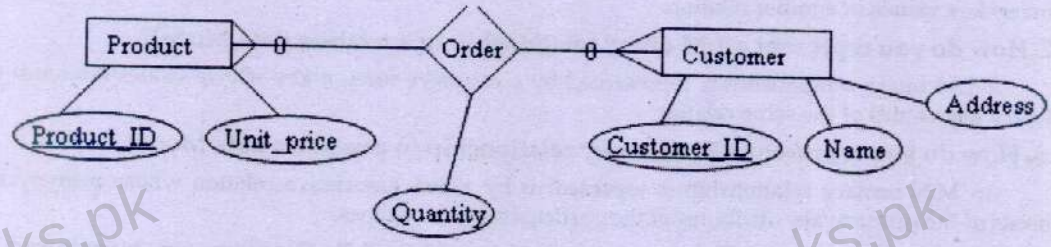

Student

| Student_ID | Name |
|---|---|

Student_phone_Number

| Student-ID | Phone_number |
|---|---|

b)




Product

| Product-ID | Unit_price |
|---|---|

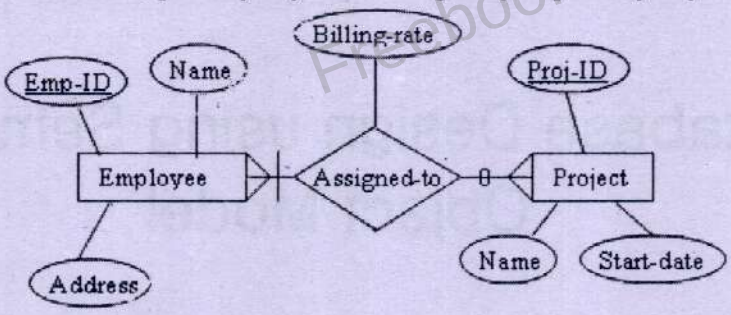
Customer

| Customer-ID | Name | Address |
|---|---|---|

Order

| Product-ID | Customer_ID | Quantity |
|---|---|---|

**Q.6. Transform the following ER diagrams into a relational schema showing the relation names, their attributes, primary keys and foreign key to primary key links.**




Employee

| Employee-ID | Name | Address |
|---|---|---|

Assigned-to

| Employee-ID | Proj-ID | Billing-Rate |
|---|---|---|

Project

| Proj-ID | Name | Start_date |
|---|---|---|

## True / False

1. Each entity type is transformed to a simple relation.
2. An M:N ternary relationship is represented by a new associative relation whose primary key consists of the primary key attributes of the participating entity types.
3. If an entity contains composite attribute, each part of the attribute is represented by a separate field in the relation
4. In 1:M relationships, the entity on the one-side of the relationship becomes a foreign key in the table of the many-side of the relationship.
5. Ternary relationship exists between the instances of three entity types
6. Unary relationship exists between the instances of different entity types
7. If an entity contains multi-valued attribute, the attribute is represented in a separate relation.

## Answers

| | | | |
|---|---|---|---|
| 1. T | 2. T | 3. T | 4. T |
| 5. T | 6. F | 7. T | |

CHAPTER 7

# Database Design using Semantic Object Model

## Chapter Overview

## 7.1 Introduction

Semantic object model is a graphical representation of a system. It is also used to describe the elements of a system and their relationships with one another. Semantic object model represents different things as **objects**. This model can be used to design the database using relational model.

## 7.2 Mapping Simple Objects

A simple object is a semantic object that contains only single-valued, non-object attributes. For example, the following figure shows a simple object COMPUTER. This object contains four attributes but none of the attribute is multi-valued and none is an object attribute. So this object is called a simple object.

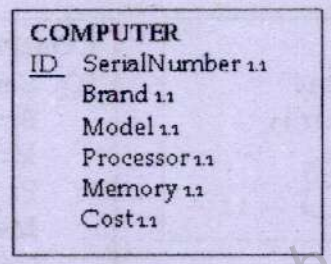



**Figure:** Simple Object COMPUTER

The mapping of simple objects is very simple. The attributes of object become the fields of relation. The attribute that identifies the object becomes the key of the relation. In semantic object model, identifier attribute is represented with underlined ID. If semantic object model contains no identifier, it should be created by adding a new attribute in relation or combining two or more existing attributes. The above simple object will be mapped as follows:

**COMPUTER** (SerialNumber, Brand, Model, Processor, Memory, Cost)

**Note:** The underlined field indicates that the field is used as primary key.

## 7.3 Mapping Composite Objects

A composite object is a semantic object that contains one or more multi-valued, non-object attributes. For example, the following figure shows a composite object EMPLOYEE. This object contains a multi-valued attribute Review so it is called a composite object.

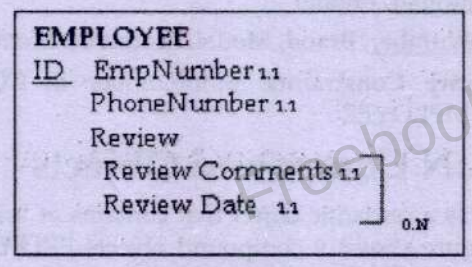



**Figure:** Composite Object EMPLOYEE

While mapping a composite object, one relation is created for the base object and an additional relation is created for repeating group attribute. The above composite object **EMPLOYEE** contains a repeating group attribute **Review**. So it will be mapped into two relations.

**EMPLOYEE** (EmpNumber, Phone)

**REVIEW** (*EmpNumber*, ReviewDate, ReviewComments)

- **Referential Integrity Constraints:** EmpNumber in REVIEW must exist in EmpNubmer in EMPLOYEE

**Note:** The underlined field(s) indicates that the field is used as primary key. The italic field(s) indicates that the field is used as foreign key.

## 7.4 Converting 1:1 Compound Objects

A compound object is a semantic object that contains at least one object attribute. The following compound objects are related with 1:1 relationship. It means that one employee can have only one computer and one computer can be assigned to only one employee.

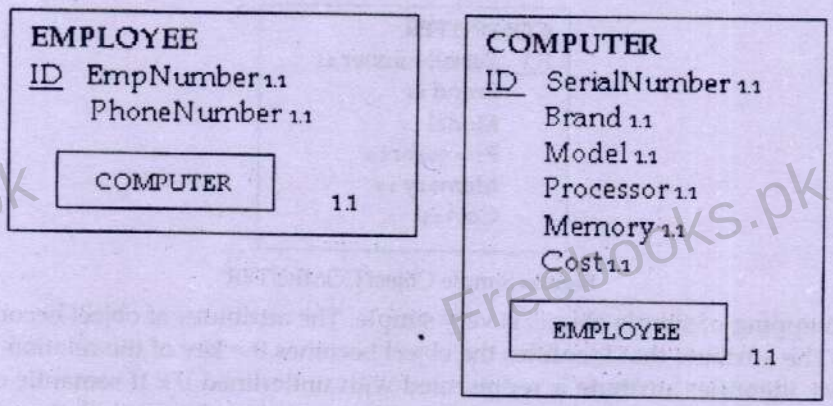



**Figure:** 1:1 Compound Object STUDENT

**EMPLOYEE** (EmpNumber, Phone, *SerialNumber*)

**COMPUTER** (SerialNumber, Brand, Model, Processor, Memory, Cost)

- **Referential Integrity Constraints:** SerialNumber in EMPLOYEE must exist in SerialNumber in COMPUTER

Another way to create two relations is as follows:

**EMPLOYEE** (EmpNumber, Phone)

**COMPUTER** (SerialNumber, Brand, Model, Processor, Memory, Cost, *EmpNumber*)

- **Referential Integrity Constraints:** EmpNumber in COMPUTER must exist in EmplNumber in EMPLOYEE

## 7.5 Converting 1:N Compound Objects

A compound object is a semantic object that contains at least one object attribute. For example, the following figure shows a compound objects PROJECT and COMPUTER. The relationship of PROJECT and COMPUTER is 1:N (One-to-Many). It means that one PROJECT may have many COMPUTERs.

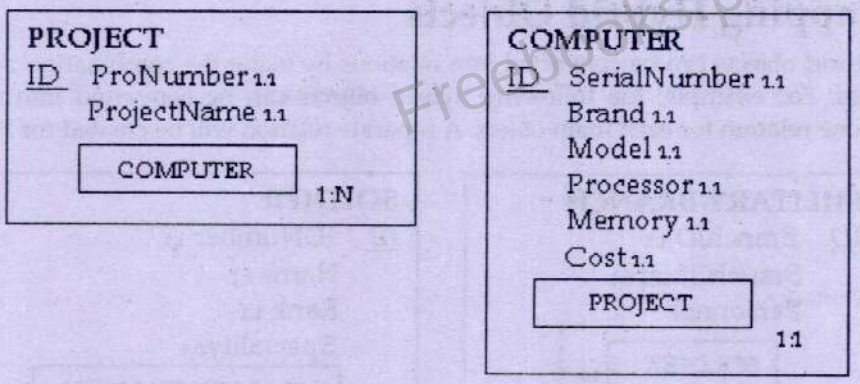



**Figure:** 1:N Compound Object

In order to convert 1:N compound object, one relation is created for the object on **one** side and one relation is created for the object on **many** side.

**PROJECT** (ProNumber, ProjectName)

**COMPUTER** (SerialNumber, Brand, Model, Processor, Memory, Cost, *ProNumber*)

- **Referential Integrity Constraints:** ProNumber in COMPUTER must exist in ProNubmer in PROJECT

## 7.6 Converting M:N Compound Objects

In order to convert M:N compound objects, we define three relations. One relation is created for each of the objects and a third relation is used as intersection relation. The intersection relation represents the relationship of the two objects and consists of the keys of both its parents.

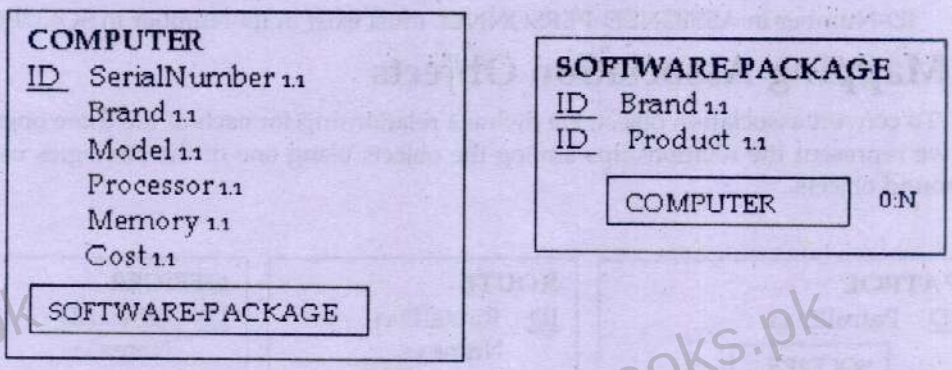



**Figure:** M:N Compound Object

**COMPUTER** (SerialNumber, Brand, Model, Processor, Memory, Cost)

**SOFTWARE-PACKAGE** (Brand, Product)

**COMPUTER-SOFTWARE-PACKAGE-INT** (*SerialNumber*, *Brand*, *Product*)

- **Referential Integrity Constraints**: SerialNumber in COMPUTER-SOFTWARE-PACKAGE-INT must exist in SerialNumber in COMPUTER
- (Brand, Product) in COMPUTER-SOFTWARE-PACKAGE-INT must exist in (Brand, Product) in COMPUTER

## 7.7 Mapping Hybrid Objects

Hybrid objects can be converted into relations by using the combination of converting techniques. For example, the following hybrid objects can be converted into relations by creating one relation for each main object. A separate relation will be created for Personnel.

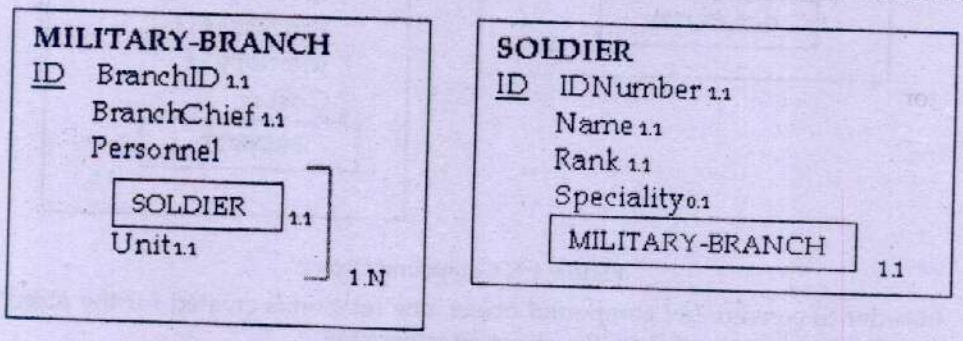



**Figure:** Hybrid object

Since a SOLDIER can only be assigned to one MILITARY-BRANCH, each instance of SOLDIER relates to only one MILITARY-BRANCH, but there are many soldiers in the Army, so SOLDIER appears many times within that one instance of MILITARY-BRANCH.

**MILITARY- BRANCH** (BranchID, BranchChief)

**SOLDIER** (IDNumber, Name, Rank, Specialty)

**PERSONNEL** (*BranchID*, *IDNumber*, Unit)

- **Referential Integrity Constraints:** BranchIdentifier in ASSIGNED-PERSONNEL must exist in BranchIdentifier in BRANCH-OF-MILITARY
- ID-Number in ASSIGNED-PERSONNEL must exist in ID-Number in SOLDIER

## 7.8 Mapping Association Objects

To convert association object, we define a relationship for each of the three objects, and then we represent the relationships among the objects using one of the strategies used with compound objects.

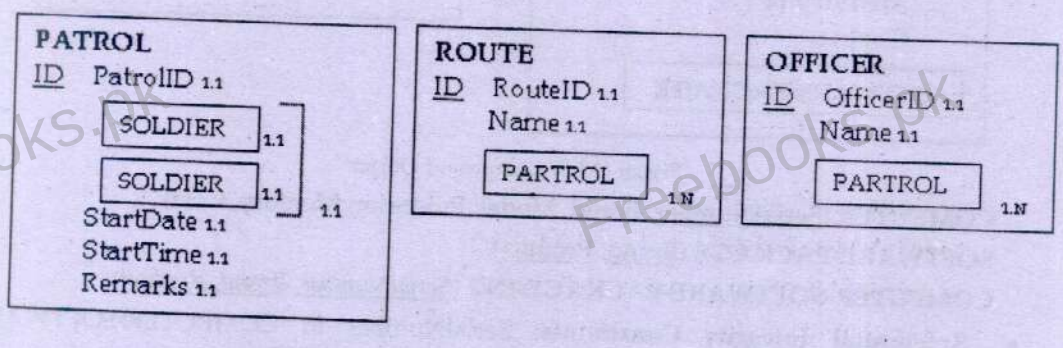



**Figure:** Association object

**PATROL** (PatrolID, *RouteID*, *OfficerID*, StartDate, StartTime, Remarks)

**ROUTE** (RouteID, Name, Route)

**OFFICER** (OfficerID, Name)

- **Referential Integrity Constraints:** RouteID in PATROL must exist in RouteID in ROUTE
- OfficerID in PATROL must exist in OfficerID in OFFICER

## 7.9 Mapping Super/Subtype Objects

To convert super/subtype objects, one relation is defined for the super object and one relation for each subtype objects. The key of each subtype relation is the key of parent relation.

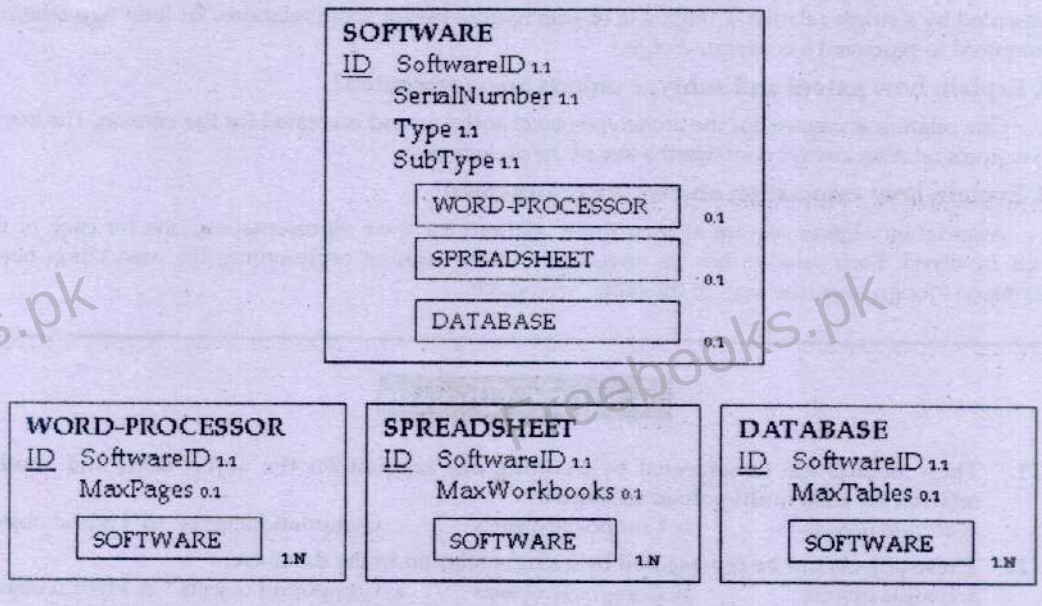



**Figure:** Relations for association object

**SOFTWARE** (SoftwareID, SerialNumber, Software-Type, Brand, Product)

**WORD-PROCESSOR** (*SoftwareID*, MaxPages)

**SPREADSHEET** (*SoftwareID*, MaxWorkbooks)

**DATABASE**(*SoftwareID*, MaxTables)

- **Referential Integrity Constraints:** SoftwareID in WORD-PROCESSOR must exist in SoftwareID in SOFTWARE-PACKAGE
- SoftwareID in SPREADSHEET must exist in SoftwareID in SOFTWARE-PACKAGE
- SoftwareID in DATABASE must exist in SoftwareID in SOFTWARE-PACKAGE

## Short Questions

**Q.1. How are hybrid objects represented?**

Hybrid objects are represented by creating a table for the multi-value group attribute of the composite object and placing by the key of the relation representing the non-composite object into that table.

**Q.2. Explain why the transformation of semantic objects into relations depends on the type of object.**

The type of object determines the number of relations and how they relate. Simple objects are represented by a single relation. Composite objects require two or more relations. At least two relations are required to represent a compound object.

**Q.3. Explain how parent and subtype objects are represented?**

One relation is created for the archetype object and a second is created for the version. The key of the version's relation always contains the key of the archetype.

**Q.4. Explain how association objects are represented?**

Association objects require at least three relations for their representation, one for each of the objects involved. Each relation has its own key, and the relation representing the association object contains, as foreign keys, the keys of the other two objects.

## Multiple Choices

1. **These objects are transformed by defining one relation for the object itself and another relation for each multi-valued attribute:**
   a. Simple objects b. Composite objects c. Compound objects d. Hybrid objects
2. **These objects can be represented by a single relation in the database:**
   a. Simple objects b. Composite objects c. Compound objects d. Hybrid objects
3. **These objects can be transformed into relational designs using a combination of the techniques for composite and compound objects.**
   a. Association objects b. Hybrid objects c. Simple objects d. Parent/sub type
4. **At least two relations are required to represent this type of object:**
   a. Simple objects b. Composite objects c. Compound objects d. Hybrid objects
5. **These objects require at least three relations for their representation:**
   a. Association objects b. Composite objects c. Compound objects d. Hybrid objects
6. **Which of the following is true about compound objects?**
   a. Only one relation is required to represent a compound object
   b. Each relation has its own distinct key
   c. At least three relations are required to represent a compound object
   d. Each relation has the same key
7. **Which of the following refers to an object that has one or more multi-value simple or group attributes but no object attributes?**
   a. Hybrid objects b. Association objects c. Simple objects d. composite objects

## Answers

| | | | |
|---|---|---|---|
| 1. b | 2. a | 3. b | 4. c |
| 5. a | 6. b | 7. d | |

## True / False

1. Simple objects can be represented by a single relation in the database.
2. Composite objects are transformed by defining one relation for the object itself and another relation for each multi-value attribute.
3. The transformation of semantic objects into relations depends on the type of object.
4. At most, two relations are required to represent a compound object.
5. There are three different types of compound objects.
6. For one-to-one relationships the key of either table is placed in the other table.
7. For one-to-many and many-to-one relationships, the key of child is placed in parent relation.
8. For many-to-many relationships, an intersection table is created that carries the keys of both relations.
9. Hybrid objects are represented by creating a table for the multi-value group attribute of the composite object and placing the key of the relation representing the composite object into that table.
10. Association objects require at least four relations for their representation, one for each of the objects involved
11. A general scheme for representing subtype objects is to create one relation for the parent and one for each of the subtypes.
12. An association object is an object that associates two other objects.
13. In general, when transforming association object structures, we only need to define one relation.
14. To represent a one-to-many relationship each object must be represented with a relation.
15. When representing a one-to-one relationship, we place the key of either relation as a foreign key in the other relation.
16. When creating relations for a compound object, each relation has its own distinct key.
17. When representing an association object, each relation has its own key, and the relation representing the association object contains, as foreign keys, the keys of the other two objects.

## Answers

| | | | |
|---|---|---|---|
| 1. T | 2. T | 3. T | 4. F |
| 5. F | 6. T | 7. F | 8. T |
| 9. F | 10. F | 11. T | 12. T |
| 13. F | 14. T | 15. T | 16. T |
| 17. T | | | |

CHAPTER 8

# Structured Query Language

## Chapter Overview

# 8.1 Structured Query Language

SQL stands for **Structured Query Language**. SQL is not a full-featured programming language. It is simply a data sublanguage. It means that it only has language statements for database definition and processing (querying and updating). The data definition commands are called **Data Definition Language (DDL)**. The data query and data updating commands are called **Data Manipulation Language (DML)**.

SQL was developed by IBM. It is endorsed as a national standard by **American National Standards Institute (ANSI)**. A newer standard SQL3 also exists but the most widely implemented version of SQL is **ANSI SQL-92** standard.

SQL works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase etc.

## 8.1.1 Features of SQL

1. SQL is an English-like language. It uses words like SELECT, INSERT, DELETE etc.
2. SQL is a non-procedural language. The user specifies what to do not how to do. SQL does not require to specify the access method to data.
3. SQL commands are not case sensitive.
4. SQL processes sets of records rather than a single record at a time. The most common form of a set of records is table.
5. SQL can be used by a range of users like DBA, application programmer, management personnel and many other types of end users.
6. SQL provides commands for a variety of tasks including:
   - Querying data
   - Inserting, updating, deleting rows in a table
   - Creating, modifying, and deleting database objects
   - Controlling access to the database and database objects
   - Guaranteeing database consistency

### Relations Used in Examples

The following three relations will be used in all examples of this chapter:

**EMP** (<u>EMPID</u>, EMPNAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)

**DEPT** (<u>DEPTNO</u>, DNAME, LOC)

**SALGRADE** (GRADE, LOSAL, HISAL)

The bold word indicates the relation name. The underline word indicates the primary key field. The remaining words in parentheses are the names of fields in the relations.

### Sample Data

The sample data in the above relations is as follows:

| DEPTNO | DNAME | LOC |
|---|---|---|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |

**Figure:** Sample data of DEPT

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1250 | 500 | 30 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1250 | 1400 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 2850 | | 30 |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 | 2450 | | 10 |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 | 3000 | | 20 |
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 | 1100 | | 20 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 950 | | 30 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | | 20 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1300 | | 10 |

**Figure:** Sample data of SALGRADE

| GRADE | LOSAL | HISAL |
|---|---|---|
| 1 | 700 | 1200 |
| 2 | 1201 | 1400 |
| 3 | 1401 | 2000 |
| 4 | 2001 | 3000 |
| 5 | 3001 | 9999 |

**Figure:** Sample data of SALGRADE

## 8.2 Basic SQL Statements

The basic SQL statements are as follows:

### 8.2.1 SELECT Statement

**SELECT** statement is used to select data from a table. It displays result in tabular form.

**Syntax**

SELECT column_name(s) FROM table_name;

**SELECT One Column**

Write a query that display EmpNo from EMP table.

SELECT EMPNO FROM EMP;

**Result**

| EMPNO |
|---|
| 7369 |
| 7499 |
| 7521 |
| 7566 |
| 7654 |
| 7698 |
| 7782 |
| 7788 |
| 7839 |
| 7844 |
| 7876 |
| 7900 |
| 7902 |
| 7934 |

## SELECT Multiple Columns

Write a query that displays the columns **EMPNO** and **ENAME** from **EMP** table.

```
SELECT EMPNO, ENAME FROM EMP;
```

### Result

```
EMPNO ENAME
----- ------
 7369 SMITH
 7499 ALLEN
 7521 WARD
 7566 JONES
 7654 MARTIN
 7698 BLAKE
 7782 CLARK
 7788 SCOTT
 7839 KING
 7844 TURNER
 7876 ADAMS
 7900 JAMES
 7902 FORD
 7934 MILLER
```

## SELECT All Columns

Write a query that displays all columns from EMP table.

```
SELECT * FROM EMP;
```

The * symbol is used instead of column names to display all columns of a table.

### Result:

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1250 | 500 | 30 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1250 | 1400 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 2850 | | 30 |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 | 2450 | | 10 |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 | 3000 | | 20 |
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 | 1100 | | 20 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 950 | | 30 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | | 20 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1300 | | 10 |

## SELECT DISTINCT Statement

**DISTINCT** keyword is used to eliminate duplicate rows from the result of a **SELECT** statement. If **DISTINCT** is not used, all rows are returned including duplicates.

## DISTINCT One Column

Write a query to displays all distinct **DEPTNO** from **EMP** table.

```
SELECT DISTINCT DEPTNO FROM EMP;
```

### Result

```
DEPTNO
-------
    10
    20
    30
```

## DISTINCT Multiple Columns

Multiple columns may be used with **DISTINCT**.

### Example

Write a query that displays distinct **DEPTNO** and **JOB** from **EMP** table.

```
SELECT DISTINCT DEPTNO, JOB FROM EMP;
```

### Result

```
DEPTNO JOB
------- ---------
    10 CLERK
    10 MANAGER
    10 PRESIDENT
    20 ANALYST
    20 CLERK
    20 MANAGER
    30 CLERK
    30 MANAGER
    30 SALESMAN
```

## SELECT Statement with WHERE Clause

**WHERE** clause is used to retrieve data from a table conditionally. It can appear only after **FROM** clause.

### Syntax

```
SELECT Column(s) FROM Table WHERE Condition;
```

### Example

Write a query that displays records of clerks from EMP table.

```
SELECT * FROM EMP WHERE JOB='CLERK';
```

### Result

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | | 20 |
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 | 1100 | | 20 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 950 | | 30 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1300 | | 10 |

## Using Quotes

SQL uses single quotes around text values. Most database systems also accept double quotes. Numeric values should not be enclosed in quotes.

## SELECT Statement with ORDER BY Clause

The **ORDER BY** clause is used to sort the rows. The process of arranging data or records in a sequence is called sorting. A sort can be ascending or descending. In ascending sort, the smallest value is placed at first position and largest position is placed at the last position. For example, 1,2,3,4,5. In descending sort, the largest value is placed at first position and the smallest value is placed at last position. For example, 5,4,3,2,1.

SQL uses **ASC** keyword to specify ascending sort and **DESC** keyword for descending sort. If neither is specified, **ASC** is used as default. **ORDER BY** must always be the last clause in **SELECT** statement. If the records contain date values, earliest date will appear first. If the records contains character values, it will be sorted alphabetically.

### Example

Write a query that displays EMP table in alphabetical order with respect to name:

```
SELECT * FROM EMP ORDER BY ENAME;
```

### Result

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7876 | ADAMS | CLERK | 7788 | 23-MAY-87 | 1100 | | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 2850 | | 30 |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 | 2450 | | 10 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | | 20 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 950 | | 30 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | | 20 |
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1250 | 1400 | 30 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1300 | | 10 |
| 7788 | SCOTT | ANALYST | 7566 | 19-APR-87 | 3000 | | 20 |
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | | 20 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1250 | 500 | 30 |

### Example

Write a query that displays **ENAME**, **JOB**, and **SAL** columns of **EMP** table in descending order by **SAL**.

```
SELECT      ENAME, JOB, SAL
FROM        EMP
ORDER BY    SAL DESC;
```

The column after **ORDER BY** clause is not required to appear in **SELECT** clause also.

### Result

| ENAME | JOB | SAL |
|---|---|---|
| KING | PRESIDENT | 5000 |
| SCOTT | ANALYST | 3000 |
| FORD | ANALYST | 3000 |
| JONES | MANAGER | 2975 |
| BLAKE | MANAGER | 2850 |
| CLARK | MANAGER | 2450 |
| ALLEN | SALESMAN | 1600 |
| TURNER | SALESMAN | 1500 |
| MILLER | CLERK | 1300 |
| WARD | SALESMAN | 1250 |
| MARTIN | SALESMAN | 1250 |
| ADAMS | CLERK | 1100 |
| JAMES | CLERK | 950 |
| SMITH | CLERK | 800 |

### Ordering by Many Columns

The ORDER BY clause can also be used with multiple columns.

#### Example

Write a query that displays name and salary of all employees from **EMP** table. Result should be sorted in ascending order by **DEPTNO** and then in descending order by **SAL**.

```
SELECT      ENAME, SAL
FROM        EMP
ORDER BY    DEPTNO, SAL DESC;
```

#### Result

```
ENAME          SAL
---------- ---------
KING          5000
CLARK         2450
MILLER        1300
SCOTT         3000
FORD          3000
JONES         2975
ADAMS         1100
SMITH          800
BLAKE         2850
ALLEN         1600
TURNER        1500
WARD          1250
MARTIN        1250
JAMES          950
```

## 8.3 Operators in SQL

Different operators in SQL are as follows:

### 8.3.1 Arithmetic Operators

Mathematical operators are the symbols that are used to perform arithmetic operations on data. An operator is a symbol that performs some operation. Arithmetic expression may contain column names, constant numeric values and the arithmetic operators. Following are the lists of available arithmetic operators in SQL.

| Operator | Symbol |
|---|---|
| + | Add |
| - | Subtract |
| * | Multiply |
| / | Divide |

#### Example

Write a query that displays ENAME, annual salary and COMM from EMP table.

```
SELECT      ENAME, SAL*12, COMM
FROM        EMP;
```

### Result

| ENAME | SAL*12 | COMM |
|---|---|---|
| SMITH | 9600 | |
| ALLEN | 19200 | 300 |
| WARD | 15000 | 500 |
| JONES | 35700 | |
| MARTIN | 15000 | 1400 |
| BLAKE | 34200 | |
| CLARK | 29400 | |
| SCOTT | 36000 | |
| KING | 60000 | |
| TURNER | 18000 | 0 |
| ADAMS | 13200 | |
| JAMES | 11400 | |
| FORD | 36000 | |
| MILLER | 15600 | |

## Operator Precedence

The order in which different types of operators in an expression are evaluated is known as **operator precedence**. It is also known as **hierarchy of operators**. Multiplication, division operators are performed before addition and subtraction. When an expression contains operators of the same order, the expression is evaluated from left to right. Use parenthesis to force a specific order of evaluation.

### Example

```
SELECT      ENAME, SAL+250*12
FROM        EMP;
```

### Result

| ENAME | SAL+250*12 |
|---|---|
| SMITH | 3800 |
| ALLEN | 4600 |
| WARD | 4250 |
| JONES | 5975 |
| MARTIN | 4250 |
| BLAKE | 5850 |
| CLARK | 5450 |
| SCOTT | 6000 |
| KING | 8000 |
| TURNER | 4500 |
| ADAMS | 4100 |
| JAMES | 3950 |
| FORD | 6000 |
| MILLER | 4300 |

In the above example, multiplication (250*12) is evaluated first. The salary value is then added to the result of multiplication.

### Example

```
SELECT      ENAME, (SAL+250)*12
FROM        EMP;
```

**Result**

```
ENAME        (SAL+250)*12
---------- ------------
SMITH             12600
ALLEN             22200
WARD              18000
JONES             38700
MARTIN            18000
BLAKE             37200
CLARK             32400
SCOTT             39000
KING              63000
TURNER            21000
ADAMS             16200
JAMES             14400
FORD              39000
MILLER            18600
```

In the above example parentheses may be used to specify the order in which operators are executed. Because of parentheses, addition takes priority over multiplication.

## 8.3.2 Relational Operators

The relational operators are used in conditions to compare one expression with another. They are used in **WHERE** clause in the following format:

WHERE exp operator value

Following are the list of Relation operators

| Operator | Meaning |
|---|---|
| > | Greater than |
| < | Less than |
| = | Equal to |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| ! = | Not equal to |

**Example**

Write a query that displays names, job and departments of all clerks.

```
SELECT    ENAME, JOB, DEPT
FROM      EMP
WHERE     JOB='CLERK';
```

**Result**

```
ENAME      JOB       DEPTNO
---------- --------- ------
SMITH      CLERK         20
ADAMS      CLERK         20
JAMES      CLERK         30
MILLER     CLERK         10
```

### Example

Write a query that displays all department names with department number greater than 20.

```
SELECT      DNAME,DEPTNO
FROM        DEPT
WHERE       DEPTNO>20;
```

### Result

```
DNAME          DEPTNO
-------------- ------
SALES              30
OPERATIONS         40
```

### Comparing One Column with Another

A column can be compared with another column in the same row as well as with a constant value.

### Example

Write a query that displays those employees whose commission is greater than their salary:

```
SELECT      ENAME,SAL, COMM
FROM        EMP
WHERE       COMM>SAL;
```

### Result

```
ENAME          SAL      COMM
---------- -------- --------
MARTIN         1250     1400
```

## 8.3.3 Logical Operators

Logical operators are used to evaluate compound conditions. The logical operators are AND, OR and NOT. AND and OR are used to give two or more conditions in a WHERE clause. NOT Negates the search condition.

- **AND:** The AND operator displays a row if ALL conditions listed are true.
- **OR:** The OR operator displays a row if ANY of the conditions listed are true.
- **NOT:** The NOT operator negates an expression.

### Example

Write a query that displays all clerks who earn between 1000 and 2000.

```
SELECT      EMPNO, ENAME, JOB, SAL
FROM        EMP
WHERE       SAL BETWEEN 1000 AND 2000
AND         JOB='CLERK';
```

### Result

| EMPNO | ENAME | JOB | SAL |
|---|---|---|---|
| 7876 | ADAMS | CLERK | 1100 |
| 7934 | MILLER | CLERK | 1300 |

### Example

Write a query that displays all employees who are either clerks and /or all employees who earn between 1000 and 2000.

```
SELECT      EMPNO, ENAME, JOB, SAL
FROM        EMP
WHERE       SAL BETWEEN 1000 AND 2000
OR          JOB='CLERK';
```

### Result

| EMPNO | ENAME | JOB | SAL |
|---|---|---|---|
| 7369 | SMITH | CLERK | 800 |
| 7499 | ALLEN | SALESMAN | 1600 |
| 7521 | WARD | SALESMAN | 1250 |
| 7654 | MARTIN | SALESMAN | 1250 |
| 7844 | TURNER | SALESMAN | 1500 |
| 7876 | ADAMS | CLERK | 1100 |
| 7900 | JAMES | CLERK | 950 |
| 7934 | MILLER | CLERK | 1300 |

The AND and OR operators can be combined in a logical expression. If AND and OR appear in same WHERE clause, all ANDs are performed first then all ORs are performed. It means that AND has a higher precedence than OR.

### Example

```
SELECT      EMPNO, ENAME, JOB, SAL
FROM        EMP
WHERE       SAL>1500
AND         JOB='MANAGER'
OR          JOB='SALESMAN';
```

### Result

| EMPNO | ENAME | JOB | SAL |
|---|---|---|---|
| 7499 | ALLEN | SALESMAN | 1600 |
| 7521 | WARD | SALESMAN | 1250 |
| 7566 | JONES | MANAGER | 2975 |
| 7654 | MARTIN | SALESMAN | 1250 |
| 7698 | BLAKE | MANAGER | 2850 |
| 7782 | CLARK | MANAGER | 2450 |
| 7844 | TURNER | SALESMAN | 1500 |

### Example

Write a query that displays all managers and salesman with salaries over 1500.

```
SELECT      EMPNO, ENAME, JOB, SAL
FROM        EMP
WHERE       SAL>1500
AND         (JOB='MANAGER'
OR          JOB='SALESMAN');
```

### Result

```
EMPNO ENAME      JOB       SAL
----- ---------- --------- ---------
 7499 ALLEN      SALESMAN     1600
 7566 JONES      MANAGER      2975
 7698 BLAKE      MANAGER      2850
 7782 CLARK      MANAGER      2450
```

The parentheses specify the order in which the operators should be evaluated. In previous example, OR operator is evaluated before AND operator.

## 8.3.4 Other Operators

Some other operators of SQL are as follows:

### BETWEEN...AND

The **BETWEEN...AND** operator retrives a range of data between two values. The values can be numbers, text or dates.

#### Syntax

```
SELECT column_name FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

#### Example

Write a query that displays those Employees whose salary is between 1000 and 2000.

```
SELECT    ENAME, SAL
FROM      EMP
WHERE     SAL BETWEEN 1000 AND 2000;
```

#### Result

```
ENAME          SAL
---------- ---------
ALLEN         1600
WARD          1250
MARTIN        1250
TURNER        1500
ADAMS         1100
MILLER        1300
```

#### Example

Write a query that displays those Employees whose salary is not between 1000 and 2000 using NOT operator.

```
SELECT    ENAME, SAL
FROM      EMP
WHERE     SAL NOT BETWEEN 1000 AND 2000;
```

#### Result

| ENAME | SAL |
|---|---|
| SMITH | 800 |
| JONES | 2975 |
| BLAKE | 2850 |
| CLARK | 2450 |
| SCOTT | 3000 |
| KING | 5000 |
| JAMES | 950 |
| FORD | 3000 |

## IN Operator

The **IN** operator is used to test for values in a specified list. It can be used with any data type.

## Example

Write a query that displays all Employees who have one of three MGR Numbers.

```
SELECT    EMPNO, ENAME, SAL, MGR
FROM      EMP
WHERE     MGR IN (8902,7566,7788);
```

## Result

| EMPNO | ENAME | SAL | MGR |
|---|---|---|---|
| 7788 | SCOTT | 3000 | 7566 |
| 7876 | ADAMS | 1100 | 7788 |
| 7902 | FORD | 3000 | 7566 |

## Example

Write a query that displays the name, job and salary for all Employees whose job is Clerk or Analyst and their salary is not equal to 1000, 3000 or 5000.

```
SELECT    ENAME,JOB, SAL
FROM      EMP
WHERE     JOB IN('CLERK','ANALYST')
AND       SAL NOT IN(1000,3000,5000);
```

## Result

| ENAME | JOB | SAL |
|---|---|---|
| SMITH | CLERK | 800 |
| ADAMS | CLERK | 1100 |
| JAMES | CLERK | 950 |
| MILLER | CLERK | 1300 |

## LIKE Operators

The **LIKE** Operator is used to specify a search for a pattern in a column. The character pattern matching operation may be referred to as "wild-card" search. Two symbols can be used to construct the search string.

- The % symbol represents any sequence of zero or more characters
- The _ symbol represents any single character

## Syntax

SELECT column FROM table WHERE column LIKE pattern

### Example

Write a query that displays all Employees whose name starts with an S.

```
SELECT      ENAME
FROM        EMP
WHERE       ENAME LIKE 'S%';
```

### Result

```
ENAME
----------
SMITH
SCOTT
```

The _ can be used to search for a specific number of characters.

### Example

Write a query that displays employee names whose names consists of four characters.

```
SELECT      ENAME
FROM        EMP
WHERE       ENAME LIKE '_____';
```

### Result

```
ENAME
----------
WARD
KING
FORD
```

### Example

Write a query that displays the names of all Employees where the third letter of their name is an A.

```
SELECT      ENAME
FROM        EMP
WHERE       ENAME LIKE '__A%';
```

### Result

```
ENAME
----------
BLAKE
CLARK
ADAMS
```

There are two underscore (_) before A in **WHERE** clause.

## IS NULL Operator

The **IS NULL** operator is used to test Null value.

### Example

Write a query that displays all Employees who have a manager (MGR).

```
SELECT      ENAME, MGR
FROM        EMP
WHERE       MGR IS NOT NULL;
```

### Result

```
ENAME          MGR
--------- ---------
SMITH        7902
ALLEN        7698
WARD         7698
JONES        7839
MARTIN       7698
BLAKE        7839
CLARK        7839
SCOTT        7566
TURNER       7698
ADAMS        7788
JAMES        7698
FORD         7566
MILLER       7782
```

## 8.4 Functions

Functions are used to manipulate data item. They accept one or more arguments and return one value. An argument is user-supplied constant, variable or column reference. The format for a function is as follows:

Function name (argument1, argument2...)

### 8.4.1 Aggregate Functions

Aggregate functions generate summary value. They can be applied to all the rows in a table or the rows specified by **WHERE** clause. Aggregate functions generate a single value from each set of rows. Aggregate functions such as SUM, COUNT, AVG, MAX, MIN generates a summary value.

#### Count (*/Column_name/Distinct Column_name)

Count function is used to count the number of rows.

#### Count (*)

The count (*) is used to count all the rows including rows containing duplicates and null values.

#### Example

Write a query that displays total columns in EMP table.

```
SELECT      COUNT (*)
FROM        EMP
WHERE       DEPTNO=20;
```

#### Result

```
COUNT(*)
---------
        5
```

The above example counts all the Employees in department 20 of EMP table including rows containing duplicate and null values.

#### Count (Column_name)

The count (Column_name) is used to count the values in the column specified excluding any null values.

### Example

Write a query that displays total records in COMM column of EMP table.

```
SELECT      COUNT (COMM)
FROM        EMP;
```

### Result

```
COUNT(COMM)
-----------
          4
```

The above example counts the values in comm. excluding null values.

### Count (Distinct Column_name)

The **COUNT** (Distinct Column_name) is used to count all the rows excluding rows containing duplicate and null values.

### Example

```
SELECT      COUNT (Distinct Mgr)
FROM        EMP;
```

### Result

```
COUNT(DISTINCTMGR)
------------------
                 6
```

The above example counts the value in **MGR** column after eliminating duplicates and null values.

## AVG Function

The **AVG** Function returns the average of all values of expression. **AVG** function can be used with numeric column only and will automatically ignore the null values

### Syntax

AVG (ALL|DISTINCT] Expression)

- **ALL:** It is the default and is applied to all values
- **DISTINCT:** It indicates that AVG is performed only on each unique instance of a value.
- **Expression:** It is any valid expression like column name.

### Example

Write a query that calculates the average salary of all employees.

```
SELECT      AVG(SAL)
FROM        EMP;
```

### Result

```
AVG(SAL)
---------
2073.2143
```

## SUM Function

The **SUM** function returns the sum of all values in an expression. It supports the use of **DISTINCT** to summarize only unique value in the expression. Null values are ignored. It can be used only with numeric columns.

### Syntax

SUM (ALL|DISTINCT] Expression)

### Example

Write a query that displays the sum of salaries of all clerks in EMP table.

```
SELECT      SUM (SAL)
FROM        EMP
WHERE       JOB='CLERK';
```

### Result

```
SUM(SAL)
---------
     4150
```

## MAX

The **MAX** function returns the maximum value in an expression. It ignores all null values. It can be used with all datatypes.

### Syntax

MAX (ALL|DISTINCT] Expression)

### Example

Write a query that finds the maximum salary earned by clerk.

```
SELECT      MAX (SAL)
FROM        EMP
WHERE       JOB='CLERK';
```

### Result

```
MAX(SAL)
---------
     1300
```

## MIN Function

The **MIN** function returns the minimum value in an expression. It ignores all null values. It can be used with all datatypes.

### Syntax

MIN (ALL|DISTINCT] Expression)

### Example

Write a query that finds the minimum salary earned by clerk.

```
SELECT      MIN (SAL)
FROM        EMP
WHERE       JOB='CLERK';
```

### Result

```
MIN(SAL)
---------
      800
```

### Example

Write a query that finds minimum, maximum and average salaries of all employees.

```
SELECT      MAX (SAL), MIN (SAL), AVG (SAL)
FROM        EMP;
```

**Result**

```
MAX(SAL)  MIN(SAL)  AVG(SAL)
--------- --------- ---------
    5000       800 2073.2143
```

## GROUP BY Clause

The **GROUP BY** clause can be used to divide the rows in a table into smaller group. If aggregate function is used in SELECT statement, GROUP BY clause produces a single value per aggregate.

### Syntax

GROUP BY Column_name

### Example

Write a query that calculates the average salary for each different job.

```
SELECT      AVG (SAL)
FROM        EMP
GROUP BY    JOB;
```

**Result**

```
 AVG(SAL)
---------
     3000
   1037.5
2758.3333
     5000
     1400
```

### Example

Write a query that finds the minimum, maximum salary for each job type.

```
SELECT      JOB, MAX (SAL), MIN (SAL)
FROM        EMP
GROUP BY    JOB;
```

**Result**

```
JOB         MAX(SAL)  MIN(SAL)
--------- --------- ---------
ANALYST        3000      3000
CLERK          1300       800
MANAGER        2975      2450
PRESIDENT      5000      5000
SALESMAN       1600      1250
```

## Excluding Rows in GROUP BY

Some rows may be pre-excluded in WHERE clause before dividing them into groups.

### Example

Write a query that displays the average salary for each Job excluding managers.

```
SELECT      JOB, AVG(SAL)
```

```
FROM        EMP
WHERE       JOB! ='MANAGER'
GROUP BY    JOB;
```

**Result**

```
JOB         AVG(SAL)
---------   ---------
ANALYST        3000
CLERK        1037.5
PRESIDENT      5000
SALESMAN       1400
```

## Groups within Groups

It is also possible to use GROUP BY clause to provide results for groups within groups.

### Example

Write a query that displays average monthly salary for each job type in department.

```
SELECT      DEPTNO, JOB, AVG (SAL)
FROM        EMP
GROUP BY    DEPTNO, JOB;
```

**Result**

```
DEPTNO JOB         AVG(SAL)
------ ---------   ---------
    10 CLERK          1300
    10 MANAGER        2450
    10 PRESIDENT      5000
    20 ANALYST        3000
    20 CLERK           950
    20 MANAGER        2975
    30 CLERK           950
    30 MANAGER        2850
    30 SALESMAN       1400
```

## HAVING Clause

The **HAVING** clause is used to filter rows after groupping them. It is also used to restrict rows by applying aggregate function in **SELECT** statement. It means that **HAVING** clause is used to restrict groups.

### Example

Write a query that displays the average salary for all departments employing more than three people:

```
SELECT      DEPTNO, AVG (SAL)
FROM        EMP
GROUP BY    DEPTNO
HAVING      COUNT (*)>3;
```

**Result**

```
 DEPTNO  AVG(SAL)
------- ---------
    20       2175
    30 1566.6667
```

### Example

Write a query that displays those jobs which have maximum salary greater than 3000:

```
SELECT      MAX (SAL), JOB
FROM        EMP
GROUP BY    JOB
HAVING      MAX (SAL)>3000;
```

### Result

```
MAX(SAL) JOB
-------- ---------
    5000 PRESIDENT
```

### Example

Write a query that finds all departments which have more than 3 Employees:

```
SELECT      DEPTNO, COUNT (*)
FROM        EMP
GROUP BY    DEPTNO
HAVING      COUNT (*)>3;
```

### Result

```
 DEPTNO  COUNT(*)
------- ---------
    20         5
    30         6
```

## Order of Clauses in SELECT Statement

The **WHERE** clause may still be used to exclude rows. The order of clauses is:

```
SELECT      column(s)
FROM        table(s)
WHERE       row condition(s)
GROUP BY    column(s)
HAVING      group of rows condition(s)
ORDER BY    columns(s);
```

SQL evaluates:

- **WHERE** clause to establish individual rows. It cannot contain group functions.
- **GROUP BY** clause to set up groups.
- **HAVING** clause to select groups for display

# 8.5 Joining

Joining is used to combine the rows from multiple tables. A join creates a temporary table with all retrieved columns from the tables specified in the join.

```
SELECT * from A, B;
```

In the above examples, A and B are two tables. It will create a temporary table with all rows from Cartesian product of table A and B. If table A contains 5 rows and table B contains 10 rows, the above statement will retrieve 5 x 10 = 50 rows. It may return duplicate or matching rows. A join requires horizontal filtering to display desired rows.

The **SELECT** statement can be written as follows:

```
SELECT * from A, B where X = Y;
```

Here, X and Y are two columns from table A and B. Table A and B must be joined with a common column between them. The tables must have at least one mathing column in order to be joined. A join between two tables should have at least one join condition between them.

There are basically three different types of joins.

- Simple Join
- Self join
- Outer Join

## 8.5.1 Simple Join

**Simple join** is the most common type of join. It retrieves rows from two tables. These tables should have a common column or set of columns that can be logically related. It is further classified into **equi-join** and **non-equi-join**.

### Equi-Join

A type of join that is based on equalities is called **equi-join**. The syntax of equi-join is as follows:

```
SELECT    table1.column, table2.column
FROM      table1, table2
WHERE     table1.column1=table2.clomumn2;
```

The condition in **WHERE** clause specifies how the tables are joined.

### Example

Write a query that displays **EMPNO, ENAME, DEPTNO, DNAME** and **LOC** from **EMP** and **DEPT** tables.

```
SELECT    EMP.EMPNO, EMP.ENAME, EMP.DEPTNO, DEPT.DNAME, DEPT.LOC
FROM      EMP, DEPT
WHERE     EMP.DEPTNO=DEPT.DEPTNO;
```

### Result

| EMPNO | ENAME | DEPTNO | DNAME | LOC |
|---|---|---|---|---|
| 7369 | SMITH | 20 | RESEARCH | DALLAS |
| 7499 | ALLEN | 30 | SALES | CHICAGO |
| 7521 | WARD | 30 | SALES | CHICAGO |
| 7566 | JONES | 20 | RESEARCH | DALLAS |
| 7654 | MARTIN | 30 | SALES | CHICAGO |
| 7698 | BLAKE | 30 | SALES | CHICAGO |
| 7782 | CLARK | 10 | ACCOUNTING | NEW YORK |
| 7788 | SCOTT | 20 | RESEARCH | DALLAS |
| 7839 | KING | 10 | ACCOUNTING | NEW YORK |
| 7844 | TURNER | 30 | SALES | CHICAGO |
| 7876 | ADAMS | 20 | RESEARCH | DALLAS |
| 7900 | JAMES | 30 | SALES | CHICAGO |
| 7902 | FORD | 20 | RESEARCH | DALLAS |
| 7934 | MILLER | 10 | ACCOUNTING | NEW YORK |

In the above example, the statement EMP.DEPTNO=DEPT.DEPTNO performs the join operation. It retrieves rows from both tables if both have the same deptno as specified in the **WHERE** clause. The **WHERE** clause used = operator to perform a join, it is an equi-join.

The column names are prefixed by table names because both tables have same column name DEPTNO. The tables names distinguish between them. If column name are unique, it is not necessary tables names.

## 8.5.2 Non-Equi-Join

A **non equi-join** specifies the relationship between columns of different tables by using relational (>, <, =, >=, <=, <>) other than =. The following example is illustrating non-equi join.

### Example

Write a query that displays ENAME, SAL and GRADE from EMP table and GRADE from SALGRADE table of those employees whose SAL of EMP table is between LOSAL and HISAL of SALGRADE table.

```
SELECT      EMP.ENAME, EMP.SAL, SALGRADE.GRADE
FROM        EMP, SALGRADE
WHERE       EMP.SAL BETWEEN SALGRADE.LOSAL AND SALGRADE.HISAL;
```

### Result

| ENAME | SAL | GRADE |
|---|---|---|
| SMITH | 800 | 1 |
| ADAMS | 1100 | 1 |
| JAMES | 950 | 1 |
| WARD | 1250 | 2 |
| MARTIN | 1250 | 2 |
| MILLER | 1300 | 2 |
| ALLEN | 1600 | 3 |
| TURNER | 1500 | 3 |
| JONES | 2975 | 4 |
| BLAKE | 2850 | 4 |
| CLARK | 2450 | 4 |
| SCOTT | 3000 | 4 |
| FORD | 3000 | 4 |
| KING | 5000 | 5 |

### Using Table Aliases

Table aliases are used to prevent ambiguity in a query. They make the queries shorter and more readable. An alias is given to the table in **FROM** clause. It is used instead of actual table name in the query.

### Example

```
SELECT      E.EMPNO, E.ENAME, E.SAL, D.DNAME, D.LOC
FROM        EMP E, DEPT D
WHERE       E.DEPTNO=D.DEPTNO;
```

### Result

Write a query that displays EMPNO, ENAME and SAL from EMP and DNAME and LOC from DEPT table. Use table alias E for EMP table and D for DEPT table.

| EMPNO | ENAME | SAL | DNAME | LOC |
|---|---|---|---|---|
| 7369 | SMITH | 800 | RESEARCH | DALLAS |
| 7499 | ALLEN | 1600 | SALES | CHICAGO |
| 7521 | WARD | 1250 | SALES | CHICAGO |
| 7566 | JONES | 2975 | RESEARCH | DALLAS |
| 7654 | MARTIN | 1250 | SALES | CHICAGO |
| 7698 | BLAKE | 2850 | SALES | CHICAGO |
| 7782 | CLARK | 2450 | ACCOUNTING | NEW YORK |
| 7788 | SCOTT | 3000 | RESEARCH | DALLAS |
| 7839 | KING | 5000 | ACCOUNTING | NEW YORK |
| 7844 | TURNER | 1500 | SALES | CHICAGO |
| 7876 | ADAMS | 1100 | RESEARCH | DALLAS |
| 7900 | JAMES | 950 | SALES | CHICAGO |
| 7902 | FORD | 3000 | RESEARCH | DALLAS |
| 7934 | MILLER | 1300 | ACCOUNTING | NEW YORK |

## 8.5.3 Self Join

Self join is a type of join in which a table is joined with itself.

### Example

Write a query that displays EMPNO, ENAME of employees along with their MGR.

```
SELECT    E.EMPNO, E.ENAME, E.MGR, M.ENAME MANAGER
FROM      EMP E, EMP M
WHERE     E.MGR=M.EMPNO;
```

### Result

| EMPNO | ENAME | MGR | MANAGER |
|---|---|---|---|
| 7369 | SMITH | 7902 | FORD |
| 7499 | ALLEN | 7698 | BLAKE |
| 7521 | WARD | 7698 | BLAKE |
| 7566 | JONES | 7839 | KING |
| 7654 | MARTIN | 7698 | BLAKE |
| 7698 | BLAKE | 7839 | KING |
| 7782 | CLARK | 7839 | KING |
| 7788 | SCOTT | 7566 | JONES |
| 7844 | TURNER | 7698 | BLAKE |
| 7876 | ADAMS | 7788 | SCOTT |
| 7900 | JAMES | 7698 | BLAKE |
| 7902 | FORD | 7566 | JONES |
| 7934 | MILLER | 7782 | CLARK |

## 8.5.4 Outer Join

The outer join extends the result of a simple join. It returns all rows returned by simple join as well as those rows from one table that do not match any row from other table. The symbol (+) represents outer join.

### Example

```
SELECT    EMP.EMPNO,EMP.ENAME, EMP.DEPTNO, DEPT.DNAME, DEPT.LOC
FROM      EMP, DEPT
WHERE     EMP.DEPTNO (+)=DEPT.DEPTNO;
```

### Result

```
EMPNO ENAME          DEPTNO DNAME          LOC
----- ---------- --------- -------------- ---------
 7782 CLARK             10 ACCOUNTING     NEW YORK
 7839 KING              10 ACCOUNTING     NEW YORK
 7934 MILLER            10 ACCOUNTING     NEW YORK
 7369 SMITH             20 RESEARCH       DALLAS
 7876 ADAMS             20 RESEARCH       DALLAS
 7902 FORD              20 RESEARCH       DALLAS
 7788 SCOTT             20 RESEARCH       DALLAS
 7566 JONES             20 RESEARCH       DALLAS
 7499 ALLEN             30 SALES          CHICAGO
 7698 BLAKE             30 SALES          CHICAGO
 7654 MARTIN            30 SALES          CHICAGO
 7900 JAMES             30 SALES          CHICAGO
 7844 TURNER            30 SALES          CHICAGO
 7521 WARD              30 SALES          CHICAGO
                           OPERATIONS     BOSTON
```

The above example retrieves rows from **DEPT** table that do not have any matching records in **EMP** table because of the presence of an outer join (+) operator. Outer join symbol (+) is used after **EMP.DEPTNO** in **WHERE** clause. It is always placed on the side which has the data deficiency.

## 8.6 Sub Query

A query within another query is called **sub query**. It is normally a **SELECT** statement inside **SELECT** statement. It is also called **nested SELECT**, **sub-SELECT** or **Inner SELECT** statement.

### Syntax

```
SELECT      Select_List
FROM        Table
WHERE       Expr Operator
            (SELECT      Select_List
            FROM         Table);
```

- The subquery (inner query) executes once before the main query
- The result of the subquery is used by the main query (outer query)

## 8.6.1 Types of Sub Query

There are two types of sub query:

### Single-Row Sub Query

Single-row sub query is a type of sub query that that returns only one row from inner SELECT statement. It uses a single row operator e.g. =,>,<,>=,<>

### Example

Write a query that displays the employee who earn the minimum salary in company. The minimum salary is unknown quantity. It requires two steps:

7. Find the minimum salary

```
SELECT MIN (SAL) FROM EMP;
```

8. Find the employee who earns the minimum salary:

```
SELECT      ENAME, JOB, SAL
FROM        EMP
WHEERE      SAL=(lowest salary which is 'unknown')
```

The above steps can be combined in a subquery as follows:

```
SELECT      ENAME, JOB, SAL      FROM  EMP
WHERE       SAL= (SELECT MIN(SAL) FROM EMP);
```

### Result

```
ENAME      JOB        SAL
---------- --------- ------
SMITH      CLERK      800
```

## Working of Subquery

A **SELECT** statement can be considered as a query block. The above example consists of two queries:

- The main query
- The inner query

The inner **SELECT** query is executed first and produces the result 800. The main query then executes and uses the value of inner query to complete search condition. The main query actually works as follows:

```
SELECT      ENAME, SAL, DEPTNO        FROM  EMP
WHERE       SAL=800;
```

In the above example, 800 is a single value. The query that returned 800 is called **single row subquery.**

## Multiple-Row Sub Query

Multiple-row sub query is a type of sub query that returns more than one row from the inner SELECT statement.

### Example

Write a query that displays the employees who earn lowest salary in each department.

```
SELECT      ENAME, SAL, DEPTNO
FROM        EMP
WHERE       SAL IN (SELECT MIN(SAL)
                    FROM EMP GROUP BY DEPTNO);
```

### Result

```
ENAME          SAL    DEPTNO
---------- --------- --------
SMITH          800       20
JAMES          950       30
MILLER        1300       10
```

The inner query has **GROUP BY** clause. It means that it may return multiple values. In this case, **IN** operator must be used because it expects a list of values.

## Comparing Multiple Values

A query can use multiple values of an inner query in **WHERE** condition.

### Example

Write a query that displays the employees who earn lowest salary in their department.

```
SELECT      ENAME, SAL, DEPTNO
FROM        EMP
WHERE       (SAL, DEPTNO) IN (SELECT    MIN (SAL), DEPTNO
             FROM EMP GROUP BY DEPTNO);
```

### Result

```
ENAME          SAL    DEPTNO
---------- --------- ---------
SMITH          800        20
JAMES          950        30
MILLER        1300        10
```

## ANY or ALL Operators

The **ANY** or **ALL** operators are used for subqueries that return multiple rows. They are used with **WHERE** or **HAVING** clause along with logical operators (=,!=,<,>,>=,<=).

### ANY Operator

The **ANY** operator compares a value to each value returned by a subquery.

- '>ANY' means more than the minimum
- '=ANY' is equivalent to IN
- '<ANY' means less than the maximum

### Example

Write a query that displays employees who earn more than the lowest salary in department 30.

```
SELECT    ENAME, SAL, JOB, DEPTNO
FROM      EMP
WHERE     SAL>ANY (SELECT DISTINCT SAL
                   FROM EMP WHERE DEPTNO=30);
```

### Result

```
ENAME          SAL JOB          DEPTNO
---------- --------- --------- ---------
ALLEN         1600 SALESMAN         30
WARD          1250 SALESMAN         30
JONES         2975 MANAGER          20
MARTIN        1250 SALESMAN         30
BLAKE         2850 MANAGER          30
CLARK         2450 MANAGER          10
SCOTT         3000 ANALYST          20
KING          5000 PRESIDENT        10
TURNER        1500 SALESMAN         30
ADAMS         1100 CLERK            20
FORD          3000 ANALYST          20
MILLER        1300 CLERK            10
```

The **DISTINCT** clause is frequently used with **ANY** operator to prevent rows from being selected several times.

### ALL Operator

The **ALL** operator compares the value to every value return by a sub query.

### Example

Write a query that displays employees who earn more than every employee in Department 30.

```
SELECT    ENAME, SAL, JOB, DEPTNO
FROM      EMP
WHERE     SAL>ALL (SELECT DISTINCT SAL
                   FROM EMP WHERE DEPTNO=30);
```

### Result

| ENAME | SAL | JOB | DEPTNO |
|---|---|---|---|
| JONES | 2975 | MANAGER | 20 |
| SCOTT | 3000 | ANALYST | 20 |
| KING | 5000 | PRESIDENT | 10 |
| FORD | 3000 | ANALYST | 20 |

## HAVING Clause with Subquery

Nested subqueries can also be used in **HAVING** clause. The **WHERE** refers to single rows and **HAVING** refers to groups of rows specified in **GROUP BY** clause.

### Example

Write a query that displays the departments which have an average salary greater than the average salary of Dept 30.

```
SELECT    DEPTNO, AVG (SAL) FROM EMP
HAVING    AVG (SAL) > (SELECT AVG (SAL)
                       FROM EMP WHERE DEPTNO = 30)
                       GROUP BY DEPTNO;
```

### Result

| DEPTNO | AVG(SAL) |
|---|---|
| 10 | 2916.6667 |
| 20 | 2175 |

### Example

Write a query that displays the job with the highest average salary:

```
SELECT JOB, AVG (SAL)
FROM EMP
GROUP BY JOB
HAVING AVG (SAL) = (SELECT MAX (AVG (SAL))
                    FROM EMP
                    GROUP BY JOB);
```

### Result

| JOB | AVG(SAL) |
|---|---|
| PRESIDENT | 5000 |

The inner query first finds the average salary for each job group and **MAX** function gets the highest average salary. The value 5000 is used in **HAVING** clause. The **GROUP BY** clause in main query is needed because main query's SELECT list contains both an aggregrate and non-aggregate column.

## Guidelines for Sub Queries

Some important points about using sub queries are as follows:

- The inner query must be in parentheses on the right hand side of the condition.

- The subquery may not have an **ORDER BY** clause.
- The **ORDER BY** clause appears at the end of the main **SELECT** statement.
- Multiple columns on **SELECT** list of inner query must be in the same order as the columns appearing on the condition clause of main query. The data type and number of columns listed must also correspond.
- Subqueries are always executed from the most deeply nested to the least deeply nested unless they are correlated subqueries (discussed later).
- Logical and SQL operators may be used as well as **ANY** and **ALL**.

## 8.7 Correlated Sub-Queries

A correlated subquery is a subquery which is executed once for each **candidate row** considered by the main query. It uses a value from a column in the outer query. It causes the correlated subquery to be processed in a different way from the ordinary nested subquery.

In normal subquery, the inner **SELECT** runs first and executes once. It returns values to main query. A correlated subquery executes once for each row considered by the outer query. The inner query is driven by the outer query. The following steps are executed for a correlated subquery:

1. Get candidate row
2. Execute inner query using candidate row's value.
3. Use value resulting from inner query to qualify or disqualify candidate.
4. Repeat until no candidate row remains.

### Example

Write a correlated subquery to find EMPloyees who earn a salary greater than the average salary for their department:

```
SELECT      EMPNO, ENAME, SAL, DEPTNO
FROM        EMP E
WHERE       SAL > (SELECT AVG (SAL)
                   FROM EMP
                   WHERE DEPTNO = E.DEPTNO);
```

### Result

| EMPNO | ENAME | SAL | DEPTNO |
|---|---|---|---|
| 7499 | ALLEN | 1600 | 30 |
| 7566 | JONES | 2975 | 20 |
| 7698 | BLAKE | 2850 | 30 |
| 7788 | SCOTT | 3000 | 20 |
| 7839 | KING | 5000 | 10 |
| 7902 | FORD | 3000 | 20 |

The alias is necessary only to avoid ambiguity in column names.

### How it Works?

The above correlated subquery works as follows:

### Main Query

1. It retrieves the first candidate row of employee Smith in DEPT 20 earning 800.
2. **EMP** in **FROM** clause has alias **E** that qualifies DEPTNO column referenced in the inner query's **WHERE** clause.
3. **WHERE** clause compares 800 against the value returned by inner query.

## Inner Query

1. It computes average salary for employee's department
2. The **WHERE DEPTNO** value is the value (E.DEPTNO) that is the value passed to inner query from outer query **DEPTNO** column.
3. **AVG (SAL)** for Smith DEPT 20 is 2175.
4. The candidate row does not meet condition and is discarded.
5. The same steps are repeated for next candidate rows of outer query.
6. The selection of candidate rows continues with those meeting the condition appearing in the query result.

## Use of Correlated Sub Query

The correlated subquery is used to read every row in the table. It compares values in each row against related data. It is used whenever a subquery must return a different result or set of results for each candidate row considered by the main query. A correlated subquery is used to answer multi-part questions whose answer depends on the value in each row of the parent query. The inner **SELECT** is normally executed once for each candidate row.

## EXISTS Operator

The **EXISTS** operator is frequently used with correlated subqueries. It checks for the existence of rows returned by the inner queries. The inner query does not return a column values rather it returns the value of true or false.

### Example

Write a query that displays employees who have at least one person reporting to them.

```
SELECT    EMPNO, ENAME, JOB, DEPTNO
FROM      EMP E
WHERE     EXISTS (SELECT EMPNO
                  FROM EMP
                  WHERE EMP.MGR = E.EMPNO)
                  ORDER BY EMPNO;
```

### Result

| EMPNO | ENAME | JOB | DEPTNO |
|---|---|---|---|
| 7566 | JONES | MANAGER | 20 |
| 7698 | BLAKE | MANAGER | 30 |
| 7782 | CLARK | MANAGER | 10 |
| 7788 | SCOTT | ANALYST | 20 |
| 7839 | KING | PRESIDENT | 10 |
| 7902 | FORD | ANALYST | 20 |

### Example

Write a correlated subquery that displays all Employees whose department is not in DEPT table.

```
SELECT    EMPNO, ENAME, DEPTNO
FROM      EMP
WHERE     NOT EXISTS (SELECT DEPTNO
                      FROM DEPT
                      WHERE DEPT. DEPTNO = EMP. DEPTNO);
```

The above query will return no records

## 8.8 Data Definition and Modification

SQL is also used for data definition and modification purposes.

### Creating Tables

The **CREATE TABLE** statement is used to create a new table. Its syntax is as follows:

```
CREATE TABLE table_name
(column1 data type, column2 data type, column3 data type);
```

The data type specifies the type of data to be stored in the column. SQL provides the following common data types:

| Data Type | Description |
|---|---|
| Char(size) | It is a fixed-length character string. The size is specified in parenthesis. The maximum size is 255 bytes. |
| Varchar (size) | It is a variable-length character string. The size is specified in parenthesis. |
| Number (size) | It is a number value. The max number of column digits is specified in parenthesis. |
| Date | It is the date value. |
| Number (size,d) | It is number value. The **size** indicates the maximum number of digits and **d** indicates the maximum number decimal points. |

### Example

```
CREATE TABLE EMP
(EMPNO      NUMBER(4) ,
ENAME       VARCHAR2(30),
JOB         VARCHAR2(15),
MGR         NUMBER(4),
SAL         NUMBER(7,2),
COMM        NUMBER(7,2),
DEPTNO      NUMBER(4));
```

### Altering an Existing Table

The **ALTER TABLE** statement is used to add or drop columns in an existing table. It is also used to change the data type of an existing field of the table. The syntax for adding a new column to an existing table is as follows:

```
ALTER TABLE table_name
ADD column_name datatype;
```

The syntax to delete an existing column from the table is as follows:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

The syntax to modify the data type of an existing column of the table is as follows:

```
ALTER TABLE table_name
MODIFY column_name data_type;
```

Some database systems do not allow to drop a column from table.

### Example

Write a query to add a column CITY in EMP table.

```
ALTER TABLE EMP ADD CITY VARCHAR(30);
```

### Example

Write a query to drop ENAME column in EMP table.

```
ALTER TABLE EMP DROP COLUMN ENAME;
```

### Example

Write a query to modify the data type of CITY column to CHAR in EMP table.

```
ALTER TABLE EMP MODIFY (CITY CHAR(25));
```

### Deleting a Table

The **DROP** statement is used to delete a table along with table structure, attributes and indexes. The syntax is as follows:

```
DROP TABLE table_name;
```

### Example

Write a query to delete EMP table.

```
DROP TABLE EMP;
```

### Truncating a Table

The **TRUNCATE TABLE** statement is used to delete all data in a table. It does not delete the table structure.

The syntax is as follows:

```
TRUNCATE TABLE table_name;
```

### Example

Write a query to delete all data in EMP table.

```
TRUNCATE TABLE EMP;
```

## 8.9 Constraints

A property that can be set on a column or set of columns in a table is called **constraint**. A database program uses constraints to enforce limitations on the date entered in tables. If a constraint is violated, the command that caused the violation is terminated. The constraint can be defined at the time of creation of a table or later. T

he constraints can be applied at the following levels:

- **Column Level Constraint:** A constraint is called column-level constraint if it is defined with column definition.
- **Table Level Constraint:** A constraint is called table-level constraint if it is not defined with a specific column definition. This constraint is applied to the whole table.

### Types of Constraints

Different constrains in SQL are as follows:

### NULL/NOT NULL

It determines whether a column can be blank or not. The word **NOT NULL** means that a column cannot have a null value. User has to supply some value in the column. A null value is not same as 0, blank, zero-length string or empty string. Null means that no entry has been made. It can only be applied at column level.

### Example

```
CREATE TABLE EMP
(EMPNO      NUMBER(4) NOT NULL,
ENAME       VARCHAR2(30) NOT NULL,
JOB         VARCHAR2(15),
MGR         NUMBER(4),
SAL         NUMBER(7,2),
COMM        NUMBER(7,2),
DEPTNO      NUMBER(4) NOT NULL);
```

## UNIQUE

A **UNIQUE** constraint specifies that all values in a given column must be unique. The column must have different value in every row of the table.

It can be used to ensure that duplicate values are not entered in a column. It can be applied at column or table level.

### Example

```
CREATE TABLE DEPT
(DEPTNO NUMBER (5) CONSTRAINT enum_ukey UNIQUE,
DNAME VARCHAR2 (14),
LOC VARCHAR2 (13));
```

### Example

```
CREATE TABLE DEPT
(DEPTNO NUMBER (5),
DNAME Varchar2 (14),
LOC VARCHAR2 (13),
CONSTRAINT enum_ukey UNIQUE (DEPTNO));
```

In the above examples, enum_ukey is the name of the constraint.

## PRIMARY KEY

It is used to define a column or set of columns as primary key. It restricts duplication of rows and does not allow Null values. A primary key that consists of two or more columns is called a **composite key**. A table can have only one primary key. Primary key constraint can be defined at table level or the column level.

### Example

The following example creates a primary key at column level.

```
CREATE TABLE DEPT
(DEPTNO NUMBER (5) PRIMARY KEY,
DNAME VARCHAR2 (14) CONSTRAINT enum_ukey UNIQUE,
LOC VARCHAR2 (13))
```

### Example

The following examples creates a primary key at table level.

```
CREATE TABLE DEPT
(DEPTNO Number (5),
DNAME varchar2 (14),
LOC Varchar2 (13)),
CONSTRAINT DEPT_UK UNIQUE(DNAME),
CONSTRAINT enum_pkey PRIMARY KEY(DEPTNO));
```

## FOREIGN KEY

It is used to define a column or set of columns as foreign key. Foreign key is used to create a link between the data in two tables. The link created between two tables is based on common field. This field is a primary key in the parent table and FOREIGN key in child table. A table can have multiple foreign keys.

It is not necessary to link a FOREIGN KEY constraint only to a primary key in another table. It may also refer to any column with UNIQUE constraint in the ther table.

- **FOREIGN KEY:** is used to define a foreign key in the child table
- **REFERENCES:** identifies the table and column in parent table
- **ON DELETE CASCADE:** specifies that when the row in parent table is deleted, the corresponding rows in child table will also be deleted. If this option is not used, the rows in parent table cannot be deleted if it is referenced in child table.

### Example

```
CREATE TABLE EMP
(EMPNO      NUMBER(4) NOT NULL,
ENAME       VARCHAR2(30) NOT NULL,
JOB         VARCHAR2(15),
HIREDATE    DATE,
MGR         NUMBER(4),
SAL         NUMBER(7,2),
COMM        NUMBER(7,2),
DEPTNO      NUMBER(4) REFERENCES DEPT);
```

### Example

```
CREATE TABLE EMP
(EMPNO      NUMBER(4) NOT NULL,
ENAME       VARCHAR2(30) NOT NULL,
JOB         VARCHAR2(15),
HIREDATE    DATE,
MGR         NUMBER(4),
SAL         NUMBER(7,2),
COMM        NUMBER(7,2),
DEPTNO      NUMBER(4),
CONSTRAINT EMP_DEP_FK FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
```

The above examples show two ways of creating foreign key. In both ways, **DEPTNO** column of **EMP** table references **DEPTNO** column of **DEPT** table.

## CHECK Constraint

The **CHECK** constraint is used to apply a test on data values in a column. It enforces domain integrity by limiting the value stored in column.

### Example

```
CREATE TABLE EMP
(EMPNO      NUMBER(10),
NAME        CHAR(20),
DEPTNO      NUMBER(2),
SALARY      NUMBER(7,2)   CHECK (SALARY<100000));
```

The above example applies a CHECK constraint on SALARY field. The value in this must be less than 100000.

### Adding or Dropping a Constraint

SQL also provides the facility to add, drop, enable or disable an existing constraint. But it is not possible to modify an existing constraint. The syntax is as follows:

```
ALTER TABLE Table_Name
ADD    [Constraint name] type (Column);
```

### Example

The following example creates a foreign key constraint on EMP table. The constraint ensures that a manager exists as valid employee in EMP table.

```
ALTER TABLE EMP
ADD CONSTRAINT EMP_FK FOREIGN KEY(MGR) REFERENCES EMP(EMPNO);
```

### Example

The following example removes the primary key constraint on DEPT table and drops the associated FOREIGN KEY constraint on EMP.DEPT column.

```
ALTER TABLE DEPT
DROP  PRIMARY KEY CASCADE;
```

### Example

The following example deletes the manager constraint from EMP table.

```
ALTER TABLE EMP
DROP  CONSTRAINT EMP_FK ;
```

## 8.10 Data Manipulation Language

The part of SQL that is used to to manipulate data in tables is known as data **manipulation language (DML).**

## The INSERT INTO Statement

The **INSERT INTO** statement is used to insert new rows in a table. The syntax of this statement is as follows:

```
INSERT INTO table_name
VALUES (value1, value2,....)
```

The column names can also be specified for which data is to be inserted:

```
INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2,....)
```

### Example

```
INSERT INTO DEPT
VALUES (50, 'EDUCATION, 'LAHORE')
```

### Example

```
INSERT INTO DEPT(DEPTNO, DNAME)
VALUES ('60', 'MIS')
```

## The UPDATE Statement

The **UPDATE** statement is used to modify the existing data in a table. The syntax of using this statement is as follows:

```
UPDATE table_name
SET column_name = new_value    WHERE column_name = some_value;
```

### Update One Column in a Row

The following statement will modify DEPTNO of SMITH:

```
UPDATE EMP SET DEPTNO = 70
WHERE ENAME = 'SMITH';
```

### Update Several Columns in a Row

The following statement will modify JOB and ENAME of SMITH:

```
UPDATE EMP
SET JOB = 'SALESMAN', ENAME = 'CLERK'
WHERE ENAME = 'SMITH'
```

## The DELETE Statement

The **DELETE** statement is used to delete rows in a table. The syntax of this statement is as follows:

```
DELETE FROM table_name
WHERE column_name = some_value
```

### Delete a Row

The following example delete SMITH:

```
DELETE FROM EMP WHERE ENAME = 'SMITH';
```

### Delete All Rows

It is possible to delete all rows in a table without deleting the table. The following statement will delete all data in the table but table structure, attributes and indexes will not be deleted:

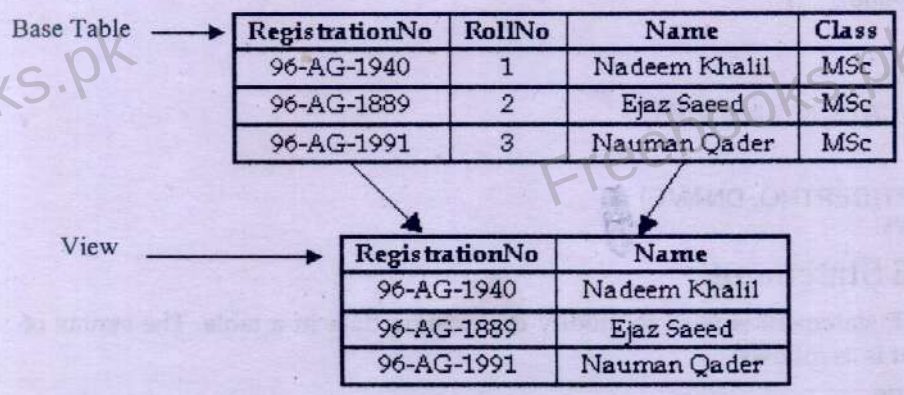
```
DELETE FROM table_name;    or    DELETE * FROM table_name;
```

### Example

```
DELETE FROM EMP;
```

## 8.11 Views

A **view** is a virtual table. Suppose a user wants to view a few columns of a base table instead of all columns. A view can be created to fulfill this request. The view consists of only those columns of the base table that the user requires. This format can be saved as a view by giving it a name.




| RegistrationNo | RollNo | Name | Class |
|---|---|---|---|
| 96-AG-1940 | 1 | Nadeem Khalil | MSc |
| 96-AG-1889 | 2 | Ejaz Saeed | MSc |
| 96-AG-1991 | 3 | Nauman Qader | MSc |

| RegistrationNo | Name |
|---|---|
| 96-AG-1940 | Nadeem Khalil |
| 96-AG-1889 | Ejaz Saeed |
| 96-AG-1991 | Nauman Qader |

**Figure:** Base table & view

### 8.11.1 Types of Views

There are two types of views:

#### 1. Simple View

Simple view is based on one table. It does not contain group functions or grouped data. Data manipulation is always possible in simple views.

**Example**

The following statements create a simple view based on EMP table:

```
CREATE OR REPLACE VIEW my_emp
AS
SELECT EMPNO, ENAME, SAL, HIREDATE, DEPTID
FROM EMP;
```

#### 2. Complex View

Complex view is based on one or more tables. It may contain group functions or grouped data. Data manipulation is not always possible in complex view.

**Example**

The following example creates a complex view based on EMP table:

```
CREATE OR REPLACE VIEW dept_sal
AS
SELECT DEPTNO, AVG(SAL) AVGSALARY
FROM EMP
GROUP BY DEPTNO;
```

#### Retrieving Data from View

The method of retrieving data from a view is similar to the method of retrieving data from a table. The following statement retrieves data from the above view:

```
SELECT * FROM dept_sal;
```

---

## Short Questions

**Q.1. What is SQL?**

SQL stands for Structured Query Language. SQL is not a full-featured programming language. It is simply a data sublanguage. It means that it only has language statements for database definition and processing (querying and updating). The data definition commands are called Data Definition Language (DDL). The data query and data updating commands are called Data Manipulation Language (DML). SQL was developed by IBM. It works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase etc.

**Q2. The questions refer to the three relations:**

**SALESPERSON** (Name, PercentofQuota, Salary)
**ORDER** (Number, CustName, SalespersonName, Amount)
**CUSTOMER** (Name, City, IndustryType)

**Sample data of SalesPerson:**

| Name | Age | Salary |
|---|---|---|
| Abdullah | 63 | 120000 |
| Babar | 38 | 42000 |
| Junaid | 26 | 36000 |
| Mahmood | 42 | 50000 |
| Zaman | 59 | 118000 |
| Kurshid | 27 | 34000 |

**Sample data of Order:**

| Number | CustName | SalespersonName | Amount |
|---|---|---|---|
| 100 | AjmalConstruction | Zaman | 560 |
| 200 | AjmalConstruction | Junid | 1800 |
| 300 | MahmoodCo | Abdullah | 480 |
| 400 | AliConstruction | Abdullah | 2500 |
| 500 | AjmalConstruction | Mahmood | 6000 |
| 600 | TahirBuilders | Abdullah | 700 |
| 700 | MuhammadCo | Junid | 150 |

**Sample Data of Customer:**

| Name | City | IndustryType |
|---|---|---|
| AjmalConstruction | FSD | B |
| MahmoodCo | LHR | F |
| TahirBuilders | PINDI | B |
| AliConstruction | PINDI | B |

An instance of these relations is shown in Figure. Use the data in those tables and show the SQL statements to display or modify data as indicated in the following questions:

a) **Show the salaries of all salespeople.**
SELECT SALARY FROM SALESPERSON;

b) **Show the salaries of all salespeople but omit duplicates.**
SELECT DISTINCT SALARY FROM SALESPERSON;

c) **Show the names of all salespeople under 30 Year old.**
SELECT Name FROM SALESPERSON
WHERE Age < 30;

d) **Show the names of all salespeople who have an order with Ajmal Construction.**
SELECT SalespersonName FROM ORDER
WHERE CustName = 'Ajmal Construction';

e) **Show the names of all salespeople who earn more than $49,999 and less than $100,000.**
SELECT Name FROM SALESPERSON
WHERE Salary > 49999 AND Salary < 100000;

f) **Show the names of all salespeople with age greater than 49 and less than 60. Use the BETWEEN keyword.**
SELECT Name FROM SALESPERSON
WHERE age BETWEEN 50 AND 59;

g) **Show names of all salespeople with age greater than 49 and less than 60. Use LIKE keyword.**
SELECT Name FROM SALESPERSON
WHERE age LIKE '5_';

**h) Show the names of customers who are located in a City ending with I.**

```
SELECT Name FROM CUSTOMER
WHERE City LIKE '%I';
```

**i) Show the names and salary of all salespeople who do not have an order with Ajmal Construction, in ascending order of salary.**

```
SELECT Name, Salary FROM SALESPERSON
WHERE Name NOT IN
            (SELECT SalespersonName FROM ORDER
            WHERE CustName = 'AjmalConstruction') ORDER BY Salary;
```

**j) Compute the number of orders.**

```
SELECT COUNT(*)FROM ORDER;
```

**k) Compute the number of different customers who have an order.**

```
SELECT COUNT (DISTINCT CustName) FROM ORDER;
```

**l) Compute the average age for salespeople.**

```
SELECT AVG(Age) FROM SALESPERSON;
```

**m) Show the name of the oldest salesperson.**

```
SELECT Name FROM SALESPERSON
        WHERE Age IN (SELECT MAX(Age) FROM SALESPERSON);
```

**n) Compute the number of orders for each salesperson.**

```
SELECT SalespersonName, COUNT(*) FROM ORDER GROUP BY SalespersonName;
```

**o) Compute the number of orders for each salesperson, considering only orders for an amount exceeding 500.**

```
SELECT COUNT(*) FROM ORDER
WHERE Amount > 500
GROUP BY SalespersonName;
```

**p) Show the names and ages of salespeople who have an order with AJMAL CONSTRUCTION, in descending order of age (use a subquery).**

```
SELECT Name, Age FROM SALESPERSON
        WHERE Name IN
            (SELECT SalespersonName FROM ORDER
            WHERE CustName = 'AjmalConstruction')
            ORDER BY Age DESC;
```

**q) Show the names and age of salespeople who have an order with AJMAL CONSTRUCTION, in descending order of age (use a join).**

```
SELECT Name, Age FROM SALESPERSON, ORDER
WHERE SALESPERSON.Name = ORDER.SalespersonName
AND ORDER.CustName = 'Ajmal Construction'
ORDER BY age DESC;
```

**r) Show the quota percentage of salespeople who have an order with a customer in PINDI (use a subquery).**

```
SELECT Age FROM SALESPERSON
WHERE Name IN
            (SELECT SalespersonName FROM  ORDER
            WHERE CustName IN
                        (SELECT Name FROM CUSTOMER
                        WHERE City = 'PINDI'));
```

**s) Show the age of salespeople who have an order with a customer in PINDI(use a join).**

```
SELECT age FROM SALESPERSON, ORDER, CUSTOMER
WHERE SALESPERSON.Name = ORDER.SalespersonName
AND ORDER.CustName = CUSTOMER.Name
AND City = 'PINDI';
```

**t) Show the industry type and names of salespeople of all orders for companies in PINDI.**

```
SELECT IndustryType, AGE FROM SALESPERSON, CUSTOMER, ORDER
WHERE SALESPERSON.Name = ORDER.SalespersonName
AND ORDER.CustName = CUSTOMER.Name
AND CITY = 'PINDI';
```

**u) Show the names of salespeople who have two or more orders.**

```
SELECT SalespersonName FROM RDER
GROUP BY SalespersonName
HAVING COUNT(*) > 1;
```

**v) Show the names and quota percentages of salespeople who have two or more orders.**

```
SELECT SALESPERSON.Name, Age FROM SALESPERSON, ORDER
WHERE SALESPERSON.Name = ORDER.SalespersonName
GROUP BY SALESPERSON.Name, Age
HAVING COUNT(*) > 1;
```

**w) Show a SQL statement to insert a new row into CUSTOMER.**

```
INSERT INTO CUSTOMER VALUES ('Qamar','SAHIWAL','F');
```

**x) Show a SQL statement to insert a new name and quota percentage into SALESPERSON; assume that salary is not determined.**

```
INSERT INTO SALESPERSON (Name, Age) VALUES ('Naeem', 35);
```

**y) Show a SQL statement to insert rows into a new table, HIGH-ACHIEVER (Name, Salary), in which, to be included, a salesperson must have a salary of at least 100,000.**

```
INSERT INTO HIGH-ACHIEVER
                    (SELECT Name, age FROM SALESPERSON
                    WHERE Salary >= 100000);
```

**z) Show a SQL statement to delete customer Ajmal CONSTRUCTION.**

```
DELETE FROM CUSTOMER
WHERE NAME = 'AjmalConstruction';
```

**aa) Show a SQL statement to delete all orders for Ajmal CONSTRUCTION.**

```
DELETE FROM ORDER
WHERE CustName = 'AjmalConstruction');
```

**bb) Show a SQL statement to change the salary of salesperson JUNID to 45,000.**

```
UPDATE SALESPERSON SET Salary = 45000
WHERE NAME = 'Junid';
```

**cc) Show a SQL statement to give all salespeople a 10 percent pay increase.**

```
UPDATE SALESPERSON SET Salary = Salary * 1.1;
```

**dd) Assume that salesperson JUNID changes his name to JAMSHAD. Show the SQL statements that make the appropriate changes.**

```
UPDATE SALESPERSON SET NAME = 'Jamshid'
WHERE NAME = 'Junid';
UPDATE ORDER SET SalespersonName = 'Jamshid'
WHERE SalespersonName = 'Junid'
```

**Q.3. Write SQL statements to create the following relations with given attribute:**

STUDENT_ID(integer, primary key)
STUDENT_Name(25 characters)
FACULTY_ID(Integer, primary key)
FACULTY_NAME(25 characters)
COURSE_ID(8 characters, primary key)
COURSE_NAME(15 characters)
DATE_QUALIFIED(date)
SECTION_ID( integer, primary key)
SEMESTER(7 characters0

**Answers:**

**CREATE TABLE STUDENT**

(STUDENT_ID NUMBER NOT NULL , STUDENT_NAME VARCHAR2(25), CONSTRAINT STUDENT_PK PRIMARY KEY (STUDENT_ID));

**CREATE TABLE FACULTY**

(FACULTY_ID NUMBER NOT NULL, FACULTY_NAME VARCHAR2(25), CONSTRAINT FACULTY_PK PRIMARY KEY (FACULTY_ID));

**CREATE TABLE COURSE**

(COURSE_ID CHAR(8) NOT NULL,COURSE_NAME VARCHAR2(15), CONSTRAINT COURSE_PK PRIMARY KEY (COURSE_ID));

**CREATE TABLE SECTION**

(SECTION_ID NUMBER NOT NULL, COURSE_ID CHAR(8), CONSTRAINT SECTION_PK PRIMARY KEY(SECTION_ID), CONSTRAINT SECTION_FK FOREIGN KEY (COURSE_ID) REFERENCES COURSE (COURSE_ID));

**CREATE TABLE IS_QUALIFIED**

(FACULTY_ID NUMBER NOT NULL, COURSE_ID CHAR(8) NOT NULL, DATE_QUALIFIED DATE, CONSTRAINT IS_QUALIFIED_PK PRIMARY KEY (FACULTY_ID, COURSE_ID)) ;

**CREATE TABLE IS_REGISTERED**

(STUDENT_ID NUMBER NOT NULL, SECTION_ID NUMBER NOT NULL, SEMESTER CHAR(7), CONSTRAINT IS_REGISTERED_PK PRIMARY KEY (STUDENT_ID, SECTION_ID), CONSTRAINT IS_REGISTERED_FK1 FOREIGN KEY(STUDENT_ID) REFERENCES STUDENT(STUDENT_ID), CONSTRAINT IS_REGISTERED_FK FOREIGN KEY (SECTION_ID) REFERENCES SECTION(SECTION_ID));

**Q.4. Write SQL definition commands for each of the following queries.**

a. How would you add an attribute CLASS to the STUDENT Table?
b. How would you remove the IS_REGISTRED table?
c. How would you change the field for FACULTY_NAME from 25 characters to 40 characters?

**Answers**

a. ALTER TABLE STUDENT
ADD (CLASS VARCHAR2 (5));
b. DROP TABLE IS_REGISTERED;
c. ALTER TABLE FACULTY
MODIFY (FACULTY_NAME VARCHAR2 (40));

**Q.5. Perform the following tasks using STUDENT table:**

a. Create two different forms of INSERT command to add a student with a STUDENT_ID of 65798 and last name ejaz to the STUDENT table.
b. Now write a command that will remove ejaz from the STUDENT Table.
c. Create an SQL command that will modify the name of COURSE ISM 4212 from database to 'Introduction to Relational Database'

**Answer:**

a. INSERT INTO STUDENT (STUDENT_ID, STUDENT_NAME)
VALUES (65798,'ejaz');
INSERT INTO STUDENT VALUES (65798,'ejaz');
b. DELETE FROM STUDENT WHERE STUDENT_ID = 65798;
c. UPDATE COURSE
SET COURSE_NAME = 'Introduction to Relational Databases'
WHERE COURSE_ID = 'ISM 4212';

**Q.6. What is difference between procedural and nonprocedural languages?**

A procedural language requires the computer programmer to specify detailed steps to complete a task. A nonprocedural language requires the programmer only to specify the task for DBMS to complete. In procedural languages, the programmer specifies what to do and how to do it. In nonprocedural, the programmer specifies what to not how to do.

**Q.7. Describe the mandatory clauses in a SQL SELECT statement.**

Two mandatory clauses are SELECT and FROM. The SELECT clause retrives the columns to be included in the result. The FROM clause identifies one or more tables and/or views from which to retrieve the column data.

**Q.8. Describe the optional clauses in a SQL SELECT statement.**

Four optional clauses are WHERE, GROUP BY, HAVING and ORDER BY. The WHERE clause carries out select operation. It specifies which rows are to be selected. The GROUP BY clause organizes data into groups by one or more column names listed in SELECT clause. The HAVING clause sets conditions regarding which groups to include in result. The groups are specified by GROUP BY clause. The ORDER BY clause sorts query results by one or more columns in ascending or descending order.

**Q.9. What is constraint?**

A property that can be set on a column or set of columns in a table is called a constraint. Database program uses constraint to enforce limitations on the date entered in a particular column. If a constraint is violated, the command that caused the violation is terminated. The constraint can be defined at the time of creation of a table or later.

**Q.10. Briefly describe four types of integrity constraints.**

The NOT NULL constraint means that a row of data must have a value for the column. The UNIQUE constraint is used to enforce uniqueness for a column value. The CHECK constraint forces data values stored in a column to fall in a range of values. The PRIMARY KEY constraint is applied to the table's primary key. It may consist of a single field or a group of fields.

**Q.11. List the SQL commands to populate, manipulate and remove rows.**

The INSERT command adds a row to the table. The UPDATE command modifies an existing row. The DELETE command removes one or more rows from a table.

**Q.12. How is updating a column different from modifying it?**

The ALTER command changes the table structure but does not change the table data. The UPDATE command changes the table data but not the table structure.

**Q.13. Why is SQL called a free-form language?**

A free-form language has no rules regarding formatting such as how many words can be put on a single line or where to break a line.

**Q.14. List the rules for writing a simple SELECT query.**

The three rules are as follows:

1. Specify the column to be displayed in the result by typing the exact complete column names.
2. Separate each column name with a comma (,).
3. Specify the name of table after FROM clause.
4. Terminate the query with a semicolon (;).

**Q.15. What is base table?**

A table from which a view is derived. A view can have onre or more base tables.

**Q.16. What is the purpose of WHERE clause?**

The WHERE clause is used to specify conditions for selecting rows to be displayed in result.

**Q.17. What is the purpose of NOT logical operator?**

The NOT logical operator negates the expression that follows it. It often simplifies the WHERE clause.

**Q.18. What is the purpose of IN operator?**

The IN operator is used to specify a list of values instead of writing a large number of OR logical operators.

**Q.19. What is the purpose of BETWEEN operator?**

The BETWEEN operator is used to specify an inclusive range of values.

**Q.20. Compare and contrast the IN and BETWEEN operators.**

Both are used to compare values from a single table column against multiple values. However, the IN operator specifies a list of values and BETWEEN operator specifies an inclusive range of values.

**Q.21. What is the purpose of LIKE operator?**

The LIKE operator is used to search for data rows containing incomplete or partial character strings within a data column.

**Q.22. Describe different operators.**

The % (percent) operator represents any string of zero or more characters. The _ (underscore) character represents any single character.

**Q.23. What happens NULL value is compared to another value?**

NULL means the absence of any stored value. When NULL value is compared with another value, the results are never true. A NULL value does not match anything not even another NULL value.

**Q.24. Describe the order of evaluation for arithmetic operators.**

Multiplication and division operators are performed before addition and subtraction. When an expression contains operators of the same order, the expression is evaluated from left to right. The parenthesis are used to force a specific order of evaluation.

**Q.25. What is difference between single row and multiple row function?**

Single row functions work on single rows only and return one result per row. There are different types of single row functions. Multiple row function operates on set or group of rows and return one result per group of rows.

**Q.26. Briefly five aggregate functions.**

The COUNT function counts all non-NULL values in a column. The AVG function computes the average of a column. The SUM function calculates the total of a column. The MIN functions returns the smallest value in a column. The MAX function returns the largest value in a column.

**Q.27. Why aggregate functions cannot be used in WHERE clause?**

A WHERE clause includes or excludes rows from a table based on user-defined criteria. The aggregate function then acts on all rows or a subset of rows that satisfy the criteria specified in WHERE clause. Since the WHERE clause must execute before the aggregate function, the aggregate function cannot be included in WHERE clause.

**Q.28. Describe the purpose of GROUP BY clause.**

The GROUP BY clause is used to specify the rows to be included from a data table when aggregating information by groups.

**Q.29. Describe a scalar aggregate function.**

A scalar aggregate function produces a single value in a result table from a SELECT statement that does not include a GROUP BY clause.

**Q.30. What is the purpose of HAVING clause?**

The HAVING clause is used to filter rows through the use of aggregate functions.

**Q.31. Compare and contrast WHERE and HAVING clauses.**

The WHERE and HAVING clauses both filter rows from tables in the result based on condition. The WHERE clause filters rows before the GROUPING action. The HAVING clause filters rows after the GROUPING action.

**Q.32. Describe three optional clauses in FOREIGN KEY constraint.**

Three optional clauses in FOREIGN KEY constraint are as follows:

1. On Update (Delete) Cascade propagates any update/delete on referenced table to referencing table.

2. On Update (Delete) Restrict rejects an update or deletion of a primary key value unless no foreign key references that value.

3. On Update (Delete) Set Null sets the value of the affected foreign key columns to null.

**Q.33. What is the purpose of table alias names?**

Table alias names are used to prevent ambiguity in a query. They make the query shorter and more readable. The shorthand method for referring to a table simplifies the process of writing a query.

**Q.34. Under what circumstances are multiple columns required to join two tables?**

Multiple columns are required to join two tables when the related tables have composite primary key columns.

**Q.35. Describe an outer join operation.**

An OUTER JOIN operation allows rows from one table to appear in a result table even if there is no matching value in the table to which it is joined.

**Q.36. Describe a SELF JOIN operation.**

A SELF JOIN operation produces a result table when the relationship of interest exists among rows that are stored within a single table.

**Q.37. What is a subquery?**

A subquery is a query within a query. A subquery is the use of a SELECT statement inside one of the clauses of another SELECT statement.

**Q.38. Describe the restrictions on subqueries.**

The restrictions on subqueries are as follows:

1. The subquery SELECT clause must contain only one expression, one aggregate function or one column name.

2. The value(s) returned by a subquery must be compatible with WHERE clause of outer query.

3. The DISTINCT keyword cannot be used in subqueries that include a GROUP BY clause.

4. Subqueries cannot manipulate their results internally.

**Q.39. How can the user decides to use a join query or a subquery?**

Use a subquery when the result table displays columns from a single table. Use a join query when the result displays columns from two or more tables.

**Q.40. Describe a scalar subquery.**

A scalar subquery is an inner query that returns a result table containing a single column from a single row.

**Q.41. Describe the effect the ALL and ANY keywords have on comparison operators.**

The ALL and ANY keywords can modify a comparison operator to allow an outer query to accept multiple values from a subquery. The ALL keyword used with > operator requires that the rows in outer query have a value greater than all values returned by inner query. The ANY keyword requires that the rows have a value greater than one or more of the values returned by the inner query.

**Q.42. How is a correlated subquery different from regular subqueries?**

A regular subquery's outer query depends on values provided by the inner query. A correlated subquery's inner query depends on values provided by the outer query.

**Q.43. Describe how EXISTS operator works.**

The EXISTS operator checks for the existence of rows returned by inner query. The inner query does not return column values. It returns a value of TRUE or FALSE.

**Q.44. Where can an ORDER BY clause be used in a subquery?**

The ORDER BY clause can be used in the outer query but not in inner query.

**Q.45. Describe the three basic types of subqueries.**

1. Some Subqueries operate on lists by using IN operator or a comparison operator modified by ANY or ALL keywords. These subqueries can return a group of values.

2. Some subqueries use an unmodified comparison operator (=, <, >, <>). These subqueries must only return a single scalar value.

3. Some subqueries use EXISTS operator to test the existence of data rows satisfying specified criteria.

---

## Multiple Choice

1. **SQL is:**
   a. Procedural language b. Non-procedural language c. Not a data sublanguage d. None
2. **To form a projection with SQL:**
   a. Name the relation to be projected and list the columns to be shown
   b. The keywords SELECT and FROM are optional
   c. The columns to be obtained arte listed before the keyword SELECT
   d. The table to be used is listed before the keyword FROM
3. **To remove duplicate rows, this qualifier must be specified:**
   a. ONLY b. UNIQUE c. DISTINCT d. SINGLE
4. **Which SQL keyword is used to name a new table and describe the table's columns?**
   a. Set b.create c. Alter d. Modify
5. **Which SQL keyword is used to change the structure, properties or constraints of a table?**
   a. Set b.create c. Alter d. Modify
6. **If table PRODUCT has a column PRICE with data type Numeric (8,2), it stores values as:**
   a. 8 digits, decimal point and 2 more digits b. 6 digits, decimal point and 2 more digits
   c. 10 digits with no stored decimal point d. 8 digits with no stored decimal point
7. **If the table PRODUCT has a column PRICE that has the data type Numeric (8,2), the value 12345 will be displayed by the DBMS as:**
   a. 123.45 b. 12345 c. 12345.00 d. 123450.00
8. **Which SQL keyword is used to delete a table's structure?**
   a. Delete b. drop c. update d. Alter
9. **When the correct SQL command is used to delete a table's structure, what happens to the data in the table?**
   a. If deleted table was a parent table, data is added to the appropriate rows of the child table.
   b. If deleted table was a child table, data is added to the appropriate rows of the parent table.
   c. The data in the table is also deleted.
   d. Nothing because there was no data in the table - only an empty table can be deleted.

**10. In an SQL query, which SQL keyword is used to specify the table(s) to be used?**
a. EXISTS b. Update c. From d. Set

**11. The asterisk (*) means that:**
a. All columns of table are to be obtained b. All records meeting the criteria are returned
c. All records with partial criteria met are returned d. None of the above

**12. Which SQL keyword is used to state the condition to specifies the rows to be selected?**
a. EXISTS b. Where c. From d. Select

**13. In an SQL query, which SQL keyword is used to join two conditions that both must be true for the rows to be selected?**
a. EXISTS b. AND c. IN d. OR

**14. In an SQL query, which SQL keyword is used to determine if a column value is equal to any one of a set of values?**
a. EXISTS b. AND c. IN d. OR

**15. In an SQL query, which built-in function is used to compute the number of rows in a table?**
a. AVG b. Max c. Min d. Count

**16. In an SQL query, which built-in function is used to total numeric columns?**
a. AVG b. SUM c. Min d. Count

**17. In SQL, which built-in function is used to compute the average value of numeric columns?**
a. AVG b. MEAN c. Min d. Count

**18. In an SQL, which built-in function is used to obtain largest value of numeric columns?**
a. AVG b. MAX c. Min d. Count

**19. In SQL, which built-in function is used to obtain the smallest value of numeric columns?**
a. AVG b. MIN c. LOWEST d. Count

**20. In an SQL query, which SQL keyword is used with built-in functions to group rows that have the same value in a specified column together?**
a. GROUP BY b. ORDER BY c. SELECT d. SORT BY

**21. Which SQL keyword is used with GROUP BY to select groups meeting specified criteria?**
a. AND b. EXISTS c. HAVING d. IN

**22. Given a table with the structure: EMPLOYEE (EmpNo, Name, Salary, HireDate), which of the following is not a valid ANSI SQL command?**
a. SELECT * FROM EMPLOYEE WHERE Name LIKE 'Ja%';
b. SELECT COUNT(*) FROM EMPLOYEE WHERE Salary < 30000;
c. SELECT COUNT(EmpNo) FROM EMPLOYEE;
d. SELECT HireDate, COUNT(*) FROM EMPLOYEE WHERE Salary < 30000;

**23. In an SQL query, which SQL keyword is used to implement a subquery?**
a. GROUP BY b. HAVING c. ORDER BY d. SELECT

**24. When one SQL query is embedded in WHERE clause of another SQL query, it is called:**
a. Subset b. Joins c. WHERE Query d. Subquery

**25. In an SQL query, which SQL keyword is used to specify the names of tables to be joined?**
a. FROM b. HAVING c. JOIN d. SELECT

**26. In SQL, which SQL keyword is used to specify the condition(s) for a join operation?**
a. FROM b. HAVING c. Where d. SELECT

**27. Regarding the interchangeability of subqueries and joins**
a. Join can always be used as an alternative to a subquery, and a subquery can always be used as an alternative to a join.
b. A join can sometimes be used as an alternative to a subquery, and a subquery can sometimes be used as an alternative to a join.
c. A join can always be used as an alternative to a subquery, and a subquery can sometimes be used as an alternative to a join.

d. A join can sometimes be used as an alternative to a subquery, and subquery can always be used as an alternative to a join.

**28. Which SQL keyword is used to add one or more rows to a table?**
a. MODIFY b. INSERT c. SELECT d. INSERT

**29. Which SQL keyword is used to change one or more rows in a table?**
a. MODIFY b. INSERT c. SELECT d. UPDATE

**30. Which SQL keyword is used to change a column value?**
a. CHANGE b. INSERT c. SELECT d. SET

**31. Which keyword is used to remove one or more rows from a table?**
A. CHANGE b. Delete c. SELECT d. Erase

**32. WHERE clauses cannot refer to ranges and to partial values when:**
a. The keyword LIKE is used b. Keyword BETWEEN is used
c. The keyword IS NULL is used d. The keyword ALSO is used

**33. Which of the following are the five built-in functions provided by SQL?**
a. COUNT, SUM, AVG, MAX, MIN b. SUM, AVG, MIN, MAX, MULT
c. SUM, AVG, MULT, DIV, MIN] d. SUM, AVG, MIN, MAX, NAME

**34. The SQL keyword GROUP BY instructs the DBMS:**
a. To physically group together rows meeting the specified criteria
b. To group together those rows that have the same value of a column
c. Both A and B d. Neither A nor B

**35. A subquery must be:**
a. In bold b. In capital letters c. In parentheses d. In italics

**36. Which is true about a "join"?**
a. A join is the combination of the WHERE and SELECT clauses
b. A join is the combination of a product operation, followed by a selection, followed (usually) by a projection
c. A join is the product of the SELECT FROM WHERE clauses
d. A join can only be used for two tables

**37. Which is true of a "join"?**
a. A join can be used as an alternative way of expressing many subqueries
b. A join expression can be substituted for all subquery expressions
c. A join expression can be substituted for EXISTS and NOT EXISTS
d. Subqueries can be substituted for all joins

**38. An outer join:**
a. Is not supported by ANSI standard SQL b. Is supported by many DBMS products
c. Both A and B d. Neither A nor B

**39. EXISTS and NOT EXISTS:**
a. Are logical operators whose value is either true or false depending on the presence or absence of rows that fit the qualifying conditions
b. Only work with one or two tables
c. Can return values other than true or false if rows meet the qualifying conditions
d. None of the above

**40. When using INSERT:**
a. Rows can be modified according to criteria
b. Rows cannot be copied in mass from one table to another
c. Rows cannot be copied in mass from one table to another
d. Rows can be inserted into a table one at a time or in groups

**41. When using DELETE:**
a. Rows can be deleted one at a time or in groups b. It will not cause integrity problems
c. Rows can be deleted only one at a time d. Order of operations is not important

**42. Which is not true about modifying data?**
a. Rows can be modified one at a time or in groups
b. The keyword SET is used to change a column value
c. After SET, the name of the column to be changed and then the new value or way of computing the new value is specified
d. Mass updates are fast, easy, and cause few problems

**43. Which of the following is not an SQL statement used on a single table?**
a. SELECT with WHERE b. SELECT with GROUP BY
c. SELECT with GROUP BY and HAVING d. SELECTION

**44. Which of the following is not an operation used on two or more tables?**
a. Subqueries b. Joins c. EXISTS d. SELECT with GROUP BY & HAVING

**45. SQL commands can be classified into three types. Which is NOT an SQL command type?**
a. DDL b. DML c. DGL d. DCL

**46. Three SQL DDL CREATE commands are**
a. Schema, Base, and Table b. Base, Table, and Schema
c. Key, Base, and Table d. Schema, Table, and View.

**47. Which is NOT an advantage of using a view?**
a. Simplify query commands b. Provide data security
c. Enhance programming productivity d. Decrease system overhead

**48. Which keyword is NOT included in most data retrieval statements?**
a. Select b. As c. From d. Where

## Answers

| | | | |
|---|---|---|---|
| 1. b | 2. a | 3. c | 4. b |
| 5. c | 6. d | 7. a | 8. b |
| 9. c | 10. c | 11. a | 12. b |
| 13. b | 14. c | 15. d | 16. b |
| 17. a | 18. b | 19. b | 20. a |
| 21. c | 22. d | 23. d | 24. d |
| 25. a | 26. c | 27. b | 28. d |
| 29. d | 30. d | 31. b | 32. d |
| 33. a | 34. b | 35. c | 36. b |
| 37. a | 38. c | 39. a | 40. d |
| 41. a | 42. d | 43. d | 44. d |
| 45. c | 46. d | 47. d | 48. b |

## True / False

1. The single most important query language is oracle.
2. SQL is a procedural language.
3. SQL is a data sublanguage, or data access language.
4. The result of every SQL query is a relation.
5. SELECT is an SQL verb that that can be used to perform a relational algebra projection or selection, and to specify other actions.
6. SQL provides five built-in functions: COUNT, SUM, AVG, MAX, and MIN.

7. SELECT Name, COUNT (*) counts the number of name rows and displays this total in a table with a single row and single column.
8. The qualifier DISTINCT must be used in an SQL statement when we want to eliminate duplicate rows.
9. In a database, field names must be unique.
10. The format SELECT-FROM-WHERE is the fundamental structure of SQL statements.
11. The condition in a WHERE clause can refer to only one value.
12. The keyword BETWEEN can be used in a WHERE clause to refer to a range of values.
13. A table is a grouping of related records.
14. A group of related fields is called a database.
15. The Keyword LIKE can be used in a WHERE clause to refer to a range of values.
16. The keyword NULL can be used in a WHERE clause to search for null (or missing) values.
17. The rows of the result relation produced by a SELECT statement can be sorted by the values in exactly one of its columns.
18. ORDER BY can be combined with any of the SELECT statements.
19. A join is a combination of a product operation, followed by a selection, followed (usually) by a production.
20. Join expressions cannot substitute for all subquery expressions, but subquery expressions can substitute for all join expressions.
21. SQL commands can be classified into two types.
22. Use the DROP command to delete a table, schema, or view.
23. The UPDATE command is used to populate tables.
24. Expressions are mathematical manipulations of the data in a table.
25. To find a range of values, use the THROUGH keyword.
26. The DML commands are used to define a database
27. To keep duplicate values from being returned use the DISTINCT keyword.
28. An outer join includes all data of each table joined.
29. Join and subquery techniques may often be used to accomplish the same task.
30. In sub queries, you may use IN instead of =.
31. In order to use the UNION clause on multiple queries, each query must output the same number of rows.

## Answers

| | | | |
|---|---|---|---|
| 1. F | 2. F | 3. T | 4. T |
| 5. T | 6. T | 7. F | 8. T |
| 9. F | 10. T | 11. F | 12. T |
| 13. F | 14. F | 15. F | 16. F |
| 17. F | 18. T | 19. T | 20. F |
| 21. F | 22. T | 23. F | 24. T |
| 25. F | 26. F | 27. T | 28. F |
| 29. T | 30. T | 31. T | |

# Concurrency & Security

## Chapter Overview

# 9.1 Transaction

A **transaction** is defined as a logical unit of work. It consists of one or more operations on a database that must be completed together so that the database remains in a consistent state. A transaction must either complete successfully or fail.

Suppose the information of stock is stored in the database. The database stores two things i.e. sold list and quantity. When an item is sold to a customer, it is added to the sold list. This is one operation on the database. However, the quantity of the sold item must also be subtracted from the items currently in stock. If both steps are not completed together, the database will be in an inconsistent state. The quantity of items in stock will be wrong if items are sold but not subtracted from items in stock. The quantity will be wrong if the items are subtracted from items in stock but not sold. Both of these operations together make up a single transaction and both must complete successfully or both must fail.

## 9.1.1 Commit and Rollback

A transaction is **committed** if it completes successfully and changes the data. If it fails and leaves the data unchanged, it is said that the transaction has been **rolled back**. It means that rollback is used to undo the work done in the current transaction.

## 9.1.2 Transaction Properties (ACID Properties)

A transaction must have four properties called **ACID** properties. These are as follows:

### 1. Atomicity

A transaction must be an atomic unit of work. A transaction must completely succeed or completely fail. If any statement in transaction fails, the entire transaction fails completely.

When a transaction is executed successfully it is said to be **committed**. In case of failure of a statement, all previous successful statements in transaction are **rolled back** or reverted to the previous state of consistency. An incomplete transaction does not take place.

### 2. Consistency

A transaction must leave the data in a consistent state after completion. For example, in a bank database, money should never be "created" or "deleted" without an appropriate deposit or withdrawal.

### 3. Isolation

All transactions that modify the data are isolated from each other. They do not access the same data at the same time. Transactions must have no dependence or effect on other transactions. A modifying transaction can access the data only before or after another transaction is completed.

### 4. Durability

The durability means that the modifications made by a transaction are permanent and persistent. If the system is crashed or rebooted, data should be guaranteed to be completed when the computer restarts.

# 9.2 Concurrency

Concurrency is a situation in which two or more users access the same piece of data at the same time. In a multi-user environment, concurrency occurs very commonly. In some situations, the concurrent access may arise to some serious problems.

## 9.2.1 Concurrency Problems

Different problems that may occur due to concurrency are as follows:

### 1. Lost Update Problem

The problem arises when two or more transactions update the same data concurrently. It occurs when two or more transactions select the same row and then update the row based on the value originally selected. Each transaction is unaware of other transactions. The last update overwrites updates made by the previous transactions. It results in loss of data.

### Example

Suppose there is a stock table that contains three attributes Product Code, Description and Quantity. There are two teams of staff, Team A and B. Team A is responsible for stock-in products and team B is responsible for stock-out products.

| Time | Team A | Stock Table | Team B |
|---|---|---|---|
| 9:00 | | Qty. = 100 | |
| 10:30 | Retrieve Qty.<br>Qty = 100. | | Retrieve Qty.<br>Qty. = 100 |
| 10:31 | | | Update Qty = Qty - 90 |
| 10:32 | | Qty. = 10 | |
| 10:33 | Update Qty = Qty + 30 | | |
| 10:34 | | Qty. = 130 | |

**Table:** Concurrency Problem

**Result:** Team B's update is lost at 10:33 which is overwritten by Team A's update.

### 2. Uncommitted Dependency Problem

This problem arises when two or more transactions work on the same table. One transaction retrieves or updates certain part of data before other transaction rollbacks update on the same data. The uncommitted dependency occurs when a second transaction selects a row that is being updated by another transaction. The second transaction is reading data that has not been committed yet and may be changed by the transaction that is updating the row

### Example

Suppose the stock table contains a field Quantity with value 100. The following sequence of actions may result in wrong result.

| Time | Team A | Team B |
|---|---|---|
| t1 | - | Update Qty to 150 |
| t2 | Retrieve Qty | - |
| t3 | - | Rollback |

In the above example, Team B updates the value of **Qty** to 150 at t1. Team A then retrieves this updated value at t2. Team B rollbacks the action while making **Qty** 100 again. So, the value of **Qty** retrieved by Team A becomes wrong at t3.

### 3. Inconsistent Analysis Problem

Inconsistent analysis problem occurs when a transaction reads several values from the database but another transaction updates some of them during the executing of the first transaction.

Inconsistent analysis is similar to uncommitted dependency in that another transaction is changing the data that a second transaction is reading. In inconsistent analysis, the data read by the second transaction was committed by the transaction that made the change. Inconsistent analysis involves multiple reads (two or more) of the same row and each time the information is changed by another transaction.

### Example

| *Three Accounts* | Acc-1: 40 | Acc-2: 50 | Acc-3: 30 |
|---|---|---|---|

| Time | Team A | Team B |
|---|---|---|
| t1 | Retrieve Acc-1: Sum = 40 | - |
| t2 | Retrieve Acc-2: Sum = 90 | - |
| t3 | - | Retrieve Acc-3 |
| t4 | - | Update Acc-3 = 0 |
| t5 | - | Retrieve Acc-1 |
| t6 | - | Update Acc-1 = 50 |
| t7 | - | Commit |
| t8 | Retrieve Acc-3: Sum = 90 | |

At the time t8, the value of sum is 90, which is wrong.

## 9.3 Resource Locking

One way to prevent concurrency problems is to lock the shared data. Locking ensures that the shared data can be used by one user at one time. When a user accesses the data, the second user has to wait until the first user finishes his work. Suppose there is an item in the database with a value 100. If two users try to access the item to update it, the locking mechanism will work as follows:

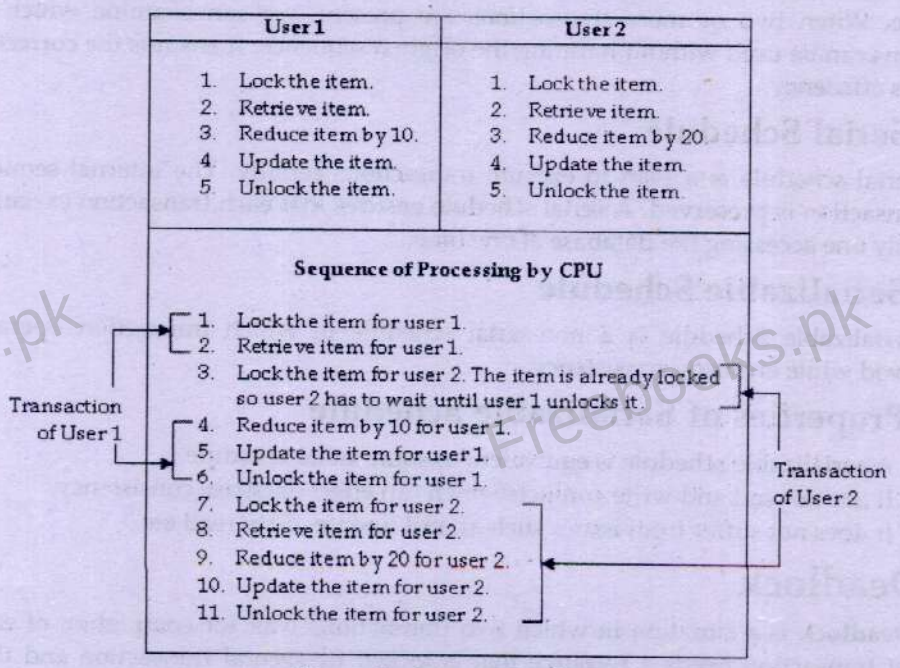



**Figure:** Locking mechanism

After both transactions are completed, the value of item will be 70. The locking mechanism ensures that if one transaction is in progress, the second transaction has to wait for the completion of first.

### 9.3.1 Lock Terminology

Some important terms related to locking mechanism are as follows:

- **Implicit Lock:** It is a type of lock placed automatically by DBMS.
- **Explicit Lock:** It is a type of lock placed by application program by issuing a command.
- **Exclusive Lock:** A type of lock that locks an item from any type of access is known as exclusive lock. If a transaction locks an item using exclusive lock, no other transaction can read or change data.
- **Shared Lock:** A type of lock that locks an item from change but not from read is known as shared lock. If a transaction has locked an item using shared lock, the other transaction can only read the item but cannot modify it.
- **Lock Granularity:** The level of the lock applied on an item is known as **lock granularity**. Different levels of locks are as follows:
    - Row level (Smallest lock granularity)
    - Page level
    - Table level
    - Database level (Largest lock granularity)

A lock with large granularity is easy to maintain for DBMS but may cause problems. A lock with small granularity is difficult to maintain for DBMS but causes less problems.

## 9.4 Serializable Transaction Schedules

A transaction consists of one or more read/write operations according to certain sequence. When two or more transactions are present, we can examine which of these operations can be used without harming the original sequence. It ensures the correctness and increases efficiency.

### 9.4.1 Serial Schedule

Serial schedule is a plan to execute transactions serially. The internal sequencing of each transaction is preserved. A serial schedule ensures that each transaction executes as if it is the only one accessing the database at one time.

### 9.4.2 Serializable Schedule

Serializable Schedule is a non-serial schedule in which transaction operations are interleaved while ensuring consistency.

### 9.4.3 Properties of Serializable Schedule

- A serializable schedule is equivalent to some serial schedule.
- It has no read and write conflicts which can effect database consistency.
- It does not suffer from issues such as, lost update, dirty read etc.

## 9.5 Deadlock

**Deadlock** is a situation in which two transactions wait for completion of each other. The first transaction needs a resource that is locked by second transaction and the second transaction needs a resource that is locked by first transaction. Deadlock is caused by locking.

Suppose there are two items A and B in the database. Two users need these items to complete their work. The following sequence of lock will result in a situation where user 1 has locked the item A and the user 2 has locked the item B. The user 1 is needs item B to complete his work. The user 2 needs item A to complete his work. Both are waiting each other. This situation is known as deadlock.

| User 1 | User 2 |
|---|---|
| 1. Lock item A.<br>2. Retrieve item A.<br>3. Lock the item B. | 1. Lock item B.<br>2. Retrieve item B.<br>3. Lock item A. |

**Sequence of Processing by CPU**

1. Lock item A for user 1.
2. Lock item B for user 2.
3. Retrieve item A for user 1.
4. Retrieve item B for user 2.
5. Put user 1 in wait for B.
6. Put user 2 in wait for A.

**Figure:** Deadlock

## 9.5.1 Solution of Deadlock

The deadlock problem can be solved in two ways:

1. Deadlock Prevention
2. Deadlock Detection & Recovery

### 1. Deadlock Prevention

There are different ways for preventing a deadlock to occur. One way is to allow users to issue only one lock request at a time. The users must lock all required resources at once. In the above example, if user 1 locks both item A and item B at once in the first line, deadlock will not occur.

### 2. Deadlock Detection & Recovery

If a deadlock occurs, it needs to be detected and then recovered. In order to recover from a deadlock situation, one transaction needs to be killed. In the above example, if the transaction of user 2 is aborted, the item B will be released from the lock and the user 1 will be able to complete his work. When a transaction is killed, all changes made by that transaction should be undone to ensure database consistency.

## 9.6 Database Failure

In every database management system, there is a possibility of hardware or software failure. The failures may occur without warning. The data in the database may be lost or damaged due to the failure of DBMS. After a failure occurs, a DBMS should recover the information that was entered into the database.

### 9.6.1 Reasons of Database Failure

The most common reasons of failure are as follows:

- Failure of computer system
- Break down of hardware
- Program bugs
- User mistakes

### 9.6.2 Types of Failures

Generally, there are two types of failures:

#### 1. System Failure

A **system failure** is also known as **instance failure.** It is a failure of the main memory of a computer system. System failures may be caused by a power failure, an application or operating system crash, memory error or some other reason. The end result is the unexpected termination of the database management system software.

#### 2. Media Failure

A **media failure** is also known as **disk failure**. It is a failure of the disk storage system of a computer system. Media failures are usually caused by head crash, fire, or exposure to vibration outside its physical operating limits.

## 9.7 Database Recovery

A database is very important for any organization. It is very important to recover it as soon as possible. Different ways for database recovery are as follows:

### 9.7.1 Recovery by Reprocessing

The simplest way to recover data is to take regular **backup** of database. If there is any failure in the database, the backup of the database can be restored. The transactions processed after the last backup are then reprocessed.

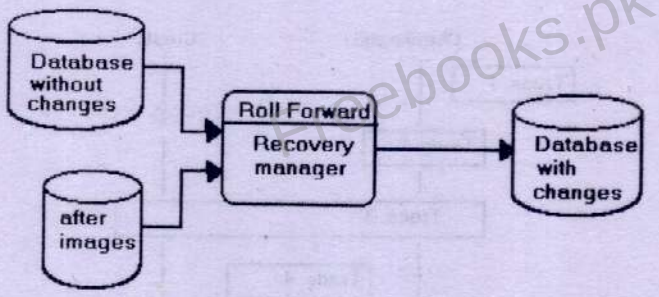
Reprocessing means to redo all events in the same way as they were done earlier. For example, if several transactions from an ATM are lost, reprocessing means going back to ATM and performing the same transactions in the same order.

Reprocessing transactions takes the same amount of time as processing them first time. It is very difficult to do if the computer is heavily scheduled. Secondly, when transactions are processed concurrently, slight variations in human activity may change the order of execution of different activities. For example, a user may insert a floppy disk more slowly or read an electronic mail message before responding to an application prompt etc.

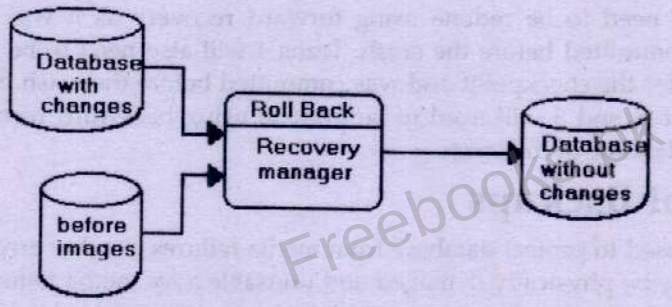
### 9.7.2 Recovery by Rollback/Rollforward

In this strategy, a transaction log file is created. The transaction logs are created to facilitate the recovery of database. All transactions that make changes to the database are recorded in transaction log before they are applied to the database. If there is any failure in the system, the log file is used to reapply all changes to the database. It is different from reprocessing as application program is not involved in this function.

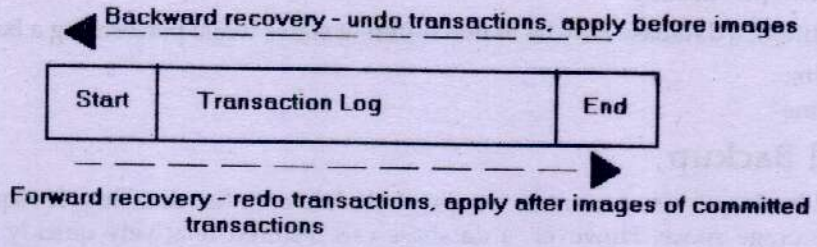
A data base that has been completely corrupted may require forward recovery from the last backup. This process would apply after images of committed transactions to the backup copy.




Secondly, the method of rollback is applied. In this process, all changes made by different transactions are undone. Then the valid transactions that were in process at the time of failure are restored. Transactions that were incomplete at the time of a failure are identified and the before image is applied to the data base.

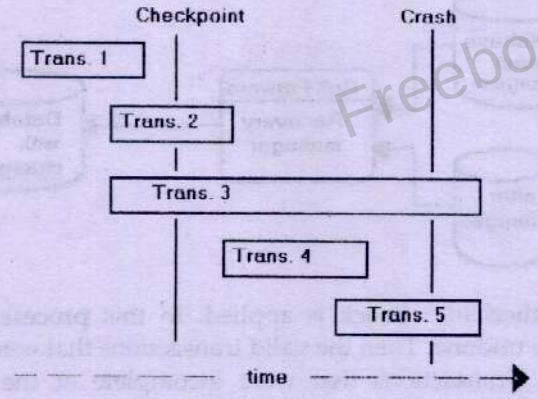



Both rollforward and rollback require log file. All changes are stored in log file before the changes are applied to the database.




Some recovery managers use strategies which involve the building of undo and redo lists. The redo list may only contain the most recent committed transactions for a given record. The undo list may only contain the earliest transaction for a given record.

Checkpoints are used in conjunction with transaction logs. A **checkpoint** is a marker that indicates the last time a database and transaction log were synchronized. If the database must be restored, only after-images for transactions that began after the last checkpoint need to be applied.




System restart will restore Trans 1 successfully that is committed before checkpoint. The Trans 2 will need to be redone using forward recovery as it was started before the checkpoint but committed before the crash. Trans 4 will also need to be forward recovered since it started after the checkpoint and was committed before the crash. So both 2 and 4 are part of a redo list. 3 and 5 will need to be undone using backward recovery as both were uncommitted at the time of the crash.

## 9.8 Types of Backups

Backup is used to protect database from media failures or other errors. One or more of database files may be physically damaged and unusable after media failure. The most recent backup can be used to replace the damaged files and reconstruct the database.

There are two main types of database backups:

1. Full backup
2. Incremental backup

In addition, a database may be in two different states while performing a backup:

1. Online
2. Offline

### 9.8.1 Full Backup

A **full backup** is used to create a copy of all data of entire database. It requires a large amount of storage space. However, a database can restored relatively quickly using a full backup as it requires to simply copy the backup files.

### 9.8.2 Incremental Backup

An **incremental backup** creates a copy of only the data that have changed since the last backup. Since an incremental backup only contains changes made to the database, the user must perform a full backup before the incremental database to restore the database at a later time. Incremental backup requires small amount of storage space. However, it may take more time to restore a database.

### 9.8.3 Offline Backups

An **offline backup** is a backup that is performed after a database has been shut down. The database administrator must schedule a time to shut down the database and notify all users so they can disconnect from the database.

Offline backup can be inconvenient for users as they must remember to complete all active transactions and disconnect from the database before shutting down. Offline backup is performed by using the commands of operating system. DBMS does not provide this facility as offline backup is performed when DBMS has been shut down.

### 9.8.4 Online Backups

An **online backup** is a backup that can be performed while a database is running. The database administrator does not have to shut down the database. The users do not need to disconnect. Online backups are more convenient for users as they perform no action.

Online backup is performed by using the commands of DBMS. The DBMS should provide the facility for online backup as DBMS is running when online backup is performed.

## 9.9 Database Security

**Database security** is a process of protecting database from intentional or accidental threats. It includes the security of hardware, software, people and data related to a database. The appropriate controls must be used to implement security effectively. The basic purpose of database security is to minimize the losses that may occur due to different security threats.

A database is an important resource of an organation that must be secured properly. Some important factors related to the security of database are as follows:

- **Theft and Fraud:** Theft and fraud affect the database environment as well as entire organization. Theft or fraud may not alter the data in database. The security checks must be applied in database environment so no person may commit such activities.
- **Loss of Confidentiality:** Confidentiality is a requirement to keep data secret. A problem in confidentiality may result in loss of competitiveness.
- **Loss of Privacy:** Privacy is a requirement to protect people's personal data. A problem in privacy may result in legal actions against the organization.
- **Loss of Integrity:** A loss in data integrity may corrupt the data or make it invalid. It may disturb the operations of the organizations seriously.
- **Loss of Availability:** It means that the data or the system may not be accessible. It can result in serious damage to the financial performance of the organization.

### 9.9.1 Database Security Threats

A **database security threat** is a situation or event that may affect a system adversaly. These threats may occur intentially or unintentionally. A threat may be a result of an event that involves a person or action to harm the system. The harm may result in tangible damage such as loss of hardware, software or data etc. It may also result in intangible damage such as loss of credibility or customer's confidence. An organization must identify all possible threat that may damage it.

### 9.9.2 Measures against Threats

Different measures can be taken against database security threats. The measures include physical controls and administrative procedures. Some measures are as follows:

#### 1. Authorization

**Authorization** is a process of granting a right to access a system or system's object. The authorization controls can be built in the software. It determines the level of access for a user to different objects in the database. Authorization controls are also known as **access controls**.

## 2. Authentication

**Authentication** is a process of determining the identity of the user. It is performed by the system administration by creating users accounts. Each user has a unique identifier that is used by operating system to identify the user. A password is typically associated with a user to access database after authentication. Some modern techniques used for authentication include finger prints, voice recognition and eye ratina etc.

- **Privileges:** The privileges are granted to the users to perform the required tasks. An unnecessary privilege may create security problems in the system. It is very critical to assign only the required privileges to each user.
- **Ownership and Privileges:** A user who creates an object becomes the owner of that object. The owner can assign appropriate privileges for the object.

## 3. Views

A **view** is a virtual table. It is the dynamic result of one or more relational operations on the relations to produce another relation. It does not actually exist in the database. The view provides powerful security mechanism by hiding the parts of database from certain users. The user is not aware of the existence of any attributes or rows missing from the view.

## 4. Backup and Recovery

**Backup** is a process of taking a copy of the database to offline storage media. DBMS also provides the facility to recover the backup in case of system failure. It is very critical to take regular backup and store it at secure location. A **log file** is also used to recover the data. This file contains the current state of transactions and database changes to provide support for recovery procedures.

## 5. Integrity

Integrity constraints are used to maintain a secure database system. These constraints prevent data from becoming invalid.

## 6. Encryption

**Encryption** is a process of encoding data by specific algorithm to make it unreadable by any program without decryption key. The encryption becomes necessary if the database contains sensitive data. Some DBMSs provide encryption facility to secure data. Encryption also protects data when transmitted over communication lines. There are many techniques for encrypting data.

## Short Questions

**Q.1 Describe five problems for organizations that create and use multi-user databases.**

1. Multi-user databases are complicated to design and develop because they support many overlapping user views.
2. Requirements change over time. The changes necessitate other changes to the database structure. Such structure changes must be carefully planned and controlled so that a change made for one group does not cause problems for another.
3. When users process a database concurrently, special controls are needed to ensure that the actions of one user do not inappropriately influence the results for another.
4. Processing rights and responsibilities need to be defined and enforced.
5. Effective backup and recovery plans, techniques and procedures are essential and complicated.

**Q.2. Define an atomic transaction and explain why atomicity is important.**

An atomic transaction is a set of actions taken on database so that all of them are performed successfully or none of them are performed at all. The database remains unchanged. Such a transaction is sometimes called a logical unit of work because it is performed as a unit.

**Q.3. Explain the difference between concurrent transactions and simultaneous transactions. How many CPUs are required for simultaneous transactions**

When two transactions are being processed against a database at the same time, they are termed concurrent transactions. With concurrent transactions, two or more users access the database using a single CPU on the database server. CPU executes some instructions from one, then executes some from the other, switching back and forth between them. In simultaneous transactions, two or more CPUs are required. Such processing is possible with modern server computers.

**Q.4. Give an example of the lost update problem.**

The lost update problem occurs when two transactions attempt to update the same data simultaneously. Each transaction copies the record into its own work area, each can make changes to its own work area then rewrite their work area to the database. The second update will destroy the first update. It creates a lost update for the first transaction. A user withdraws money from an ATM, a friend withdraws money from the same account at the same time at a different ATM. It 'might' by possible that first withdrawal is destroyed when the balance is written back to database by second withdrawal.

**Q.5. Explain the difference between an explicit and an implicit lock.**

Locks placed by DBMS are called implicit locks. Locks placed by command are called explicit locks.

**Q.6. What is lock granularity?**

The size of a lock is called lock granularity. Locks with large granularity are easy for DBMS to administer but frequently cause conflicts. Locks with small granularity are difficult to administer as there are many more details for DBMS to track and check. But conflicts are less common.

**Q.7. Explain the difference between an exclusive lock and a shared lock.**

An exclusive lock locks the item from access of any type. No other transaction can read or change the data. A shared lock locks the item from change but not from read. Other transactions can read the item as long as they do not attempt to alter it.

**Q.8. Explain two-phased locking.**

Transactions are allowed to obtain locks as necessary. Once first lock is released, no other lock can be obtained. Transactions have a growing phase in which the locks are obtained. They have a shrinking phase in which the locks are released.

**Q.9. What is deadlock? How can it be avoided? How can it be resolved once it occurs?**

Deadlock occurs when User1 locks a resource needed by User2 and User2 locks a resource needed by User1. Each wait for a resource that other person has locked. One way to avoid deadlock is to issue all lock requests at one time. Users must lock all resources they want at once. When deadlock occurs, normal solution is to rollback one of the transactions to remove its changes from the database.

**Q.10. Explain the difference between optimistic and pessimistic locking.**

Optimistic locking assumes that no conflict will occur. Data is read, transaction is processed, updates are issued and then a check is made to see if conflict occurred. If not, the transaction is finished. If so, the transaction is repeated until it processes with no conflict. Pessimistic locking assumes that conflict will occur. First, locks are issued, the transaction is processed, and then the locks are freed.

**Q.11. Explain the use of BEGIN, COMMIT, and ROLLBACK TRANSACTION statements.**

BEGIN defines the beginning or starting point of a transaction. COMMIT marks the successful completion of the transaction. ROLLBACK backs out any change made by the transaction.

**Q.12. Explain the meaning of the expression ACID transaction.**

An ACID transaction is one that is atomic, consistent, isolated and durable. An atomic transaction is one in which either all of the database actions occur or none of them occur. A durable transaction is one for which all committed changes are permanent. DBMS will not lose or remove such changes even in the case of failure. If transaction is durable, DBMS will provide facilities to recover the changes of all committed actions when necessary.

**Q.13. How a database can be recovered via reprocessing. Why is it generally not feasible?**

Reprocessing means to redo all events in the same way as they were done earlier. For example, if several transactions from an ATM are lost, reprocessing means going back to ATM and performing the same transactions in the same order.

Reprocessing transactions takes the same amount of time as processing them first time. It is very difficult to do if the computer is heavily scheduled. Secondly, when transactions are processed concurrently, slight variations in human activity may change the order of execution of different activities. For example, a user may insert a floppy disk more slowly or read an electronic mail message before responding to an application prompt etc.

**Q.14. Define rollback and rollforward.**

In a rollback, we undo changes made by erroneous or partially processed transactions by undoing the changes they have made in the database. We apply before images to the changed database data. Then, the valid transactions that were in process at the time of the failure are restarted.

In a rollforward, the database is restored using the saved data, and all valid transactions since the save are reapplied. We apply after images to the restored database data.

**Q.15. Why is it important to write to the log before changing the database values?**

If system crashes between the time a transaction is logged and the time it is applied, there is a record of an unapplied transaction. If, on the other hand, the transactions were to be applied before they were logged, it would be possible to change the database but have no record of the change. If this happened, an user might reenter an already completed transaction.

**Q.16. Describe the rollback process. Under what conditions should it be used?**

In a rollback, current database and transaction log are used. Before images of all uncommitted transactions are placed back on the database and any failed transaction is restarted. A rollback is used when a transaction fails or a system failure occurs that does not damage active database.

**Q.17. Describe the rollforward process. Under what conditions should it be used?**

In a rollforward, the saved copy of the database and the transaction log are used. First, the database is restored from the saved copy. Next, after images of all committed transactions are placed back on the database and all failed transactions are restarted. A rollforward is used when a failure has occurred that renders the database unusable.

**Q.18. What is the advantage of taking frequent checkpoints of a database?**

Checkpoints are inexpensive operations. It is feasible to take three or four (or more) per hour. In this way, no more than 15 or 20 minutes of processing needs to be recovered.

**Q.19. Explain the concept of serializable transactions.**

Concurrent transactions are two or more transactions that are processed against the database at the same time. It is desirable for concurrent transactions to be serializable; that is, the results of the concurrent transactions should be logically consistent with the results that would be obtained if the transactions were not processed concurrently but rather in an arbitrary serial order.

**Q.20. Explain the purpose of transaction logs and checkpoints.**

Transaction logs are created to facilitate the recovery of database. All transactions that make changes to database are recorded in transaction log before they are applied to database. If database fails, the transactions in the log can be used to undo changes made by transactions that were not committed and to redo changes that were committed since the database was last saved. A checkpoint is a marker for when the last time the database and the transaction log were synchronized. If the database must be restored, only after-images for transactions that began after the last checkpoint have to be applied

**Q.21. Which are more commonly used – implicit or explicit locks? Why?**

Implicit locks are more commonly used than explicit locks. Concurrency control involves many complex factors that influence the performance of the system. The impact of some factors can only be determined through trial and error. Changing explicit locks to tune system performance can require making changes throughout the program code to obtain and release locks at various places in the transactions. Implicit locks are much easier to change since a locking strategy can be specified in a system parameter or lock declaration area and then the DBMS will place the locks implicitly wherever they are needed to implement that strategy.

---

## Multiple Choices

1. **The measures taken to prevent one user's work from inappropriately influencing another user's work are called:**
   a. Concurrency control b. Checkpoint c. Database recovery d. Database logging.
2. **A series of actions to be taken on the database such that either all actions are completed successfully, or none of them can be completed, is known as a(n):**
   a. Checkpoint b. Log c. Lock d. Transaction
3. **When two transactions are processed against database at the same time, they are:**
   a. Called concurrent transactions b. Usually interleaved
   c. Resulting in a lost update problem d. Both a and b
4. **The situation that occurs when one user's changes to the database are lost by a second user's changes to the database is known as:**
   a. Concurrent update problem. b. Deadly embrace problem.
   c. Inconsistent read problem. d. Inconsistent write problem.
5. **One remedy for the inconsistencies caused by concurrent processing is:**
   a. Lost updates b. Check pointing c. Rollback d. Resource locking
6. **A lock placed automatically by the DBMS is called a(n) _____ lock:**
   a. Exclusive b. Explicit c. Granular d. Implicit
7. **Which of the following is not true about locks?**
   a. Locks with large granularity are easier for the DBMS to administer.
   b. Locks with small granularity cause more conflicts.
   c. Locks with large granularity produce fewer details for the DBMS to track.
   d. Locks may have a table-level granularity.

8. **Which type of lock prevents all types of access to the locked resource?**
   a. Exclusive lock b. Shared lock c. Two-phased lock d. Explicit lock
9. **Which type of lock allows other transactions to have read-only access to locked resource?**
   a. Exclusive lock b. Shared lock c. Two-phased lock d. Explicit lock
10. **Which of the following is NOT true about two-phased locking?**
    a. Can make transactions serializable b. uses only shared locks
    c. Has a growing phase d. has a shrinking phase
11. **The situation that occurs when two users are waiting for a resource that the other person has locked is known as a(n):**
    a. Lost update problem b. Deadlock.
    c. Inconsistent read problem. d. Inconsistent write problem.
12. **Requiring all application programs to lock resources in same order prevents what problem?**
    a. Concurrent update b. Lost update c. Deadlock d. Exclusive locks
13. **Locks that are placed assuming that a conflict will occur are called:**
    a. Dynamic locks b. Explicit locks c. Implicit locks d. Pessimistic locks.
14. **Once processing rights have been defined may be implemented at any level EXCEPT:**
    a. Network b. Operating system c. Data d. DBMS
15. **Recovering a database via reprocessing involves:**
    a. Restoring database from the save and reprocessing all the transactions since the save.
    b. Restoring database from save and reapplying all changes made by transactions since save.
    c. Undoing changes made by erroneous or partially processed transactions and restarting the valid transactions that were in process at the time of the failure.
    d. Recreating database by reentering all of the data from the beginning, and then reprocessing all of the transactions.
16. **Which of the following is not true of DBMS security features?**
    a. Users may be assigned to one or more roles b. A role may be assigned to only one user
    c. Users and roles can have many permissions d. Objects have many permissions.
17. **Recovering a database via rollforward involves:**
    a. Restoring database from save and reprocessing all transactions since the save.
    b. Restoring database from save and reapplying all changes made by transactions since save.
    c. Undoing the changes made by erroneous or partially processed transactions and restarting the valid transactions that were in process at the time of the failure.
    d. Re-creating the database by re-entering all of the data from the beginning and then reprocessing all of the transactions.
18. **Recovering a database via rollback involves:**
    a. Restoring the database from the save and reprocessing all the transactions since the save.
    b. Restoring the database from the save and reapplying all the changes made by transactions since the save.
    c. Undoing the changes made by erroneous or partially processed transactions, and restarting the valid transactions that were in process at the time of the failure.
    d. Re-creating the database by re-entering all of the data from the beginning and then reprocessing all of the transactions.
19. **Which of the following would not be contained in a transaction log?**
    a. Before-images b. Type of operation c. Pointers d. Permissions
20. **Copies of each database record or page before it was changed by a transaction that are saved for use in database recovery are called**
    a. Before-images b. Type of operation c. Pointers d. Time of the action
21. **Copies of each database record or page after it was changed by a transaction that are saved for use in database recovery are called:**
    a. After-images b. Pointers c. Time of the action d. Permissions

**22. Which of the following cannot be enforced in the DBMS or application programs?**
a. Processing rights b. Security c. Processing responsibilities d. Transaction isolation

## Answers

| | | | |
|---|---|---|---|
| 1. a | 2. d | 3. d | 4. a |
| 5. d | 6. d | 7. b | 8. a |
| 9. b | 10. b | 11. c | 12. c |
| 13. d | 14. a | 15. a | 16. b |
| 17. b | 18. c | 19. d | 20. a |
| 21. d | | | |

## True/False

1. Transaction is a series of actions to be taken on the database so that either all of them are performed successfully or none of them are performed at all.
2. Concurrency control measures are taken to ensure that one user's work has absolutely no influence on another user's work.
3. A transaction is a group of alternative database actions from which the database can choose to perform only one of them.
4. "Resource locking" is one remedy to the lost update problem.
5. Resource locking must be carefully planned because most DBMS products cannot detect a deadlock condition.
6. When two transactions are being processed against the database at the same time they are termed concurrent transactions.
7. The lost update problem is when User A reads data that have been processed by a portion of a transaction from User B.
8. Resource locking is a process to prevent multiple applications from obtaining copies of the same record when the record is about to be changed.
9. Locks placed by command are implicit locks .
10. Locks placed by the DBMS are explicit locks.
11. The size of the lock is referred to as the lock granularity.
12. An exclusive lock locks the item from change but not from read.
13. The goal of concurrency control is to ensure that one user's work does not inappropriately influence another user's work.
14. Two transactions that run concurrently and generate results that are consistent with the results that would have occurred if they had run separately are referred to as serializable transactions.
15. A point of synchronization between the database and the transaction log is generally referred to as a stop point.
16. The log contains a copy of every database record (or page) after it has changed. These records are called before-images.

17. The goal of database security is to ensure that only authorized users can perform authorized activities at authorized times.
18. When one transaction reads a changed record that has not been committed to the database a filthy read occurs.
19. In regard to database security, neither the DBMS nor the database applications can enforce processing responsibilities.
20. Processing rights may be implemented at the DBMS level.
21. All commercial DBMS products use some version of "username and password" as part of their security features.
22. The security provided by the DBMS often has to be augmented by additional security features within the application program.
23. Reprocessing is normally the most convenient method for recovery after a system failure.
24. Rollforward and reprocessing are two different names for the same technique.
25. Both rollforward and rollback require the use of a log of transaction results.
26. An ACID transaction is one that is atomic, consistent, isolated, and durable.
27. A durable transaction is one in which all committed changes are permanent.

## Answers

| | | | |
|---|---|---|---|
| 1. T | 2. F | 3. F | 4. T |
| 5. F | 6. T | 7. F | 8. T |
| 9. F | 10. F | 11. T | 12. F |
| 13. T | 14. T | 15. F | 16. F |
| 17. T | 18. F | 19. T | 20. T |
| 21. T | 22. T | 23. F | 24. F |
| 25. T | 26. T | 27. T | |

CHAPTER 10

# Client–Server Databases & ODBC

## Chapter Overview

# 10.1 Client-Server Architecture

Client-server architecture is a network of computers. It consists of two types of computers known as **clients** and **servers**. The client computers process applications and server computers provide services to the client computers such as database processing.

**Client-server database** system is based on client-server architecture. In this database system, the server computer stores and processes a database. The client computers use the database by sending requests to the server computer.

## 10.1.1 Client-Server Processing

In client-server processing, application programs performs the following tasks:

- Process the user interface
- Invoke application logic
- Enforce some business rules
- Calls the DBMS for database services. The call is normally given in SQL.

The DBMS performs the following tasks:

- Processes SQL statements
- Enforce some business rules
- Returns data or error messages to the application program
- Provides reliability and security

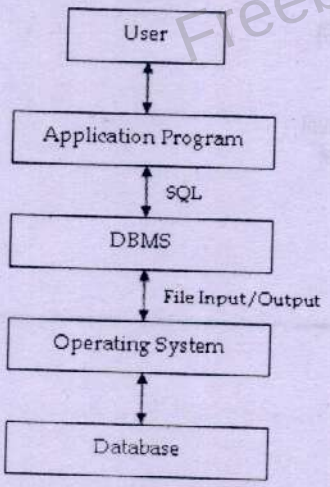



**Figure:** Programs in Database Processing

These functions are required in all types of database systems. In personal databases, all functions are performed on a single personal computer.

## 10.1.2 Functions of Client and Server

In client-server database systems, application program is stored on client computers and DBMS is placed on server computer. Another software placed on the client is known as **DBMS driver**. DBMS driver receives the processing requests from application program and sends to DBMS. It also receives responses from the DBMS and sends them to application program. The communication between client and server computers is done by using communication software. This software is placed on both client and server and enables them to send and receive messages to one another.

#### 10.1.2.1 Functions of Client Computer

The client computer performs the following tasks:

- Manages the user interface
- Accepts data from the user
- Processes application logic
- Enforces business rules
- Generates requests for database services
- Transmits requests to the server
- Receives results from the server
- Presents the results to the user

#### 10.1.2.2 Functions of Server Computer

The server computer performs the following tasks:

- Accepts the client's requests
- Processes client's requests
- Enforces rules
- Performs database integrity checks
- Provides concurrent access control
- Returns the result to the client
- Performs recovery and security services

### 10.1.3 Advantages of Client-Server System

The advantages of client-server databases are as follows:

#### 1. Concurrent Access

In client-server database system, several clients can access database simultaneously. It provides **multi-user environment** to satisfy the requests of many users at the same time.

#### 2. Better Performance

In client-server database system, several CPUs process the database application simultaneously. It results in better performance and efficient processing.

#### 3. Reduced Cost

In client-server database system, the communication costs are reduced. Only the requests for DBMS processing and their results are sent to communication network. It needs less communication traffic than file-sharing systems.

#### 4. Better User Interface

The client computers are not used to process database. The processing power of these computers can be used for better graphical user interfaces. We can use more sophisticated menus and forms with colors and graphics.

### 10.1.4 Disadvantage

In client-server database system, there is possibility of less control. In many users are accessing the server simultaneously, it can result in wrong data in the database. For example, if two users are trying the update database, the final change may be inconsistent. This is known as **concurrency problem**.

# 10.2 Reliability and Security in Client-Server Systems

Client-server system is a multi-user system in which many users can access the database at the same time. The system must ensure security and reliability of the data in the database. Security and reliability is maintained in client-server systems as follows:

## 10.2.1 Concurrency Control

Different types of problem may occur if the data is accessed simultaneously by different clients. Suppose two users are trying to update the value of a field at the same time. The value of the field is 250. Following concurrent problem may occur:

| User 1 | User 2 |
|---|---|
| Retrieve 250 from the field in variable **n** | |
| | Retrieve 250 from the field in variable **m** |
| Set n = n + 100 | |
| Update the field with **n** | |
| | Set m = m + 50 |
| | Update the field with **m** |

The final value updated in the field is 300, which is wrong. The actual value in the field should be 400. The above problem can be prevented in two ways:

- Pessimistic Locking
- Optimistic Locking

### 10.2.1.1 Pessimistic Locking

Pessimistic locking is used with each transaction automatically. Whenever a transaction starts, the DBMS locks the database. If any other user tries to access the database at this time, it is now allowed. When the transaction is completed, the database is unlocked and the other client can access the database. The sequence of execution in the above example will be as follows when pessimistic locking is used:

| User 1 | User 2 |
|---|---|
| Retrieve 250 from the field in variable **n** | |
| Set n = n + 100 | |
| Update the field with **n** | |
| | Retrieve 350 from the field in variable **m** |
| | Set m = m + 50 |
| | Update the field with **m** |

When the user 1 starts transaction, the database is locked and user 2 cannot use the database until the transaction of user 1 is complete.

### 10.2.1.2 Optimistic Locking

The optimistic locking mechanism works with an assumption that there will be no conflict. It allows transactions to proceed as if there were no concurrency problems. But before a transaction commits, a check is performed to determine whether conflict has occurred or not. In case of a conflict, the transaction is rolled back.

### 10.2.2 Recovery

If any problem occurs and data is lost, the client-server system is responsible for recovering the data. For this purpose, the server computer keeps a log of change sin the database. The database is periodically backed up to tape or another medium. When failures occur, the server DBMS recovers the data from backup or log files.

### 10.2.3 Security

The security in client-server system is maintained by different ways. The system administrator allocates usernames and passwords to users and assigns specific rights to them. Different access levels and privileges are applied on tables and views. The user needs to login using password for using the database.

Different layers of security can also be implemented. For example, the communication software may require a username and password to access the network. Secondly, the operating system on the server may again required username and password to grant access to the database file. The DBMS may again require a third username and password to grant access to different tables and views.

## 10.3 Open Database Connectivity (ODBC)

**Open Database Connectivity (ODBC)** is a standard interface that is used to process different types of databases. ODBC is DBMS-independent and process databases developed in different database management systems. It means that an application that uses ODBC interface can process Oracle database, Access database or any other database without changing any program code.

The basic purpose of ODBC is to allow a developer to develop a single application. The application can access databases supported by different DBMS without changing the code and recompiling it.

ODBC was developed by a committee of industry experts from **X/Open** and **SQL Access Group** committees. It has been implemented by Microsoft and is part of Windows. It is important in client-server database systems because the client application can process databases developed in different types of DBMS.

### 10.3.1 ODBC Architecture

There are many components of ODBC standard. The application program, driver manager, and DBMS drivers reside on the client computer. The drivers send requests to data sources. The data source resides on the server computer. The **data source** consists of database, its associated DBMS, operating system and network platform. An ODBC data source can be a relational database, a file server or a spreadsheet etc.

The application issues a request to create a connection with the data source in order to perform different operations on the database. ODBC provides a standard means for each of these requests and defines a standard set of error codes and messages.

The **driver manager** serves as an intermediary between the application and the DBMS drivers. When the application requests a connection, the driver determines the type of DBMS that processes a given ODBC data source and loads that driver in memory. The driver manager also processes some initialization requests and validates the format and order of ODBC requests that it receives from application. This driver manager is provided by Microsoft and is included in Windows.

The **driver** processes ODBC requests and submits specific SQL statements to data source. There is a different driver for each type of data source. The drivers are supplied by DBMS vendors and by independent software companies. The driver also ensures that standard ODBC commands are executed correctly. If the data source is not SQL-compatible, the driver has to convert the SQL command into such command that is understood by the data source. The driver also converts the data source error codes and messages into ODBC standard codes and messages.

ODBC identifies two types of drivers:

### 1. Single-Tier Driver

A single-tier driver processes both ODBC calls and SQL statements.

### 2. Multi-Tier Driver

A multi-tier driver processes ODBC calls but passes the SQL requests directly to the data source.

## 10.4 Conformance Levels

There are two types of conformance levels:

- ODBC Conformance Level
- SQL Conformance Level

### 10.4.1 ODBC Conformance Level

ODBC conformance level is related to the feature and functions that available driver's **application program interface (API)**. An API is a set of functions that perform a particular task. The application can call an API to receive its services.

An application can call a driver to determine ODBC conformance level. If the conformance level required by the application is not present, it can terminate the session properly and generate messages to the user. The developer can write the program to use higher ODBC conformance level if available and lower ODBC conformance level if higher level is not available.

### 10.4.2 SQL Conformance Level

SQL conformance levels specify which SQL statements, expression and data types a driver can process. The application can call the driver to determine SQL conformance level. If higher SQL conformance level is not available, the application may use lower level for obtaining the required data from the data source.

The application will call an API function to specify a particular task. The API call is passed on to the translator. The translator issues command in the language of DBMS to specify the requested action.

The translator is referred to as the ODBC Driver. It is the driver that implements the various API such that they are understood by the DBMS. Thus, there is a driver for each DBMS to be accessed.

The ODBC Driver accepts the command from the application and converts it to a format understood by the target DBMS. In addition, the ODBC Driver also receives the result of the command execution from the DBMS and passes it back to the application.

## Short Questions

**Q.1. What is Client-Server Architecture?**

Client-server architecture is a network of computers. It consists of two types of computers known as clients and servers. The client computers process applications and server computers provide services to the client computers such as database processing.

Client-server database system is based on client-server architecture. In this database system, the server computer stores and processes a database. The client computers use the database by sending requests to the server computer.

**Q.2. Name the components of the ODBC standard.**

Application program, driver manager, DBMS drivers, and data source

**Q.3. What role does the driver manager serve? Who supplies it?**

The driver manager serves as an intermediary between application and DBMS drivers. When the application requests a connection, the driver determines the type of DBMS that processes a given ODBC data source and loads that driver in memory if it is not already loaded. The driver manager also processes certain initialization requests and validates the format and order of ODBC requests that it receives from the application. For Windows, the driver manager is provided by Microsoft.

**Q.4. What role does the DBMS driver serve? Who supplies it?**

A driver processes ODBC requests and submits specific SQL statements to a given type of data source. There is a different driver for each data source type. Drivers are supplied by DBMS vendors and by independent software companies.

**Q.5. Briefly describes the components of ODBC architecture.**

In ODBC architecture, the application program, driver manager and DBMS drivers reside on the Web server. The database and DBMS reside on database server. The combination of the database and its associated DBMS comprise the data source. The data source contains data used by the application. The application program talks to driver manager. When the application requests a connection, driver manager checks to see which DBMS the data source uses and loads appropriate DBMS driver into memory. The driver manager also checks the format of ODBC requests coming from the application program. The DBMS driver processes ODBC requests and submits SQL statements to data source. The driver also ensures that the responses coming from the data source are in appropriate ODBC format.

**Q.6. What are the different types of conformance levels within ODBC standard and why do they exist?**

There are two types of conformance levels within the ODBC standard – ODBC conformance and SQL conformance. ODBC conformance deals with the features and functions that are available to the application program through DBMS driver API. SQL conformance standard deals with SQL statements, expressions and data types that the driver can process. The reason for these levels is to accommodate the varying ability of different vendors to comply with the power and expressiveness of ODBC standard and SQL language.

**Q.7. What is the difference between an interface and an implementation?**

An interface is specified by a set of objects and the properties and methods that they expose. An object need not expose all of its properties and methods in a given interface. How the object supports the interface, or the implementation, is completely hidden from the user. Developers of an object are free to change the implementation whenever they want, but they may not ever change the interface without incurring the justifiable disdain of their users.

**Q.8. What is a single-tier driver?**

A single-tier driver processes both ODBC calls and SQL statements. Used with file-server oriented DBMS products.

**Q.9. What is a multiple-tier driver?**

A multiple-tier driver processes ODBC calls but passes SQL statements to the server DBMS for processing.

**Q.10. Do the uses of the term tier in the three-tier architecture and its use in ODBC have anything to do with each other?**

No, there is no relationship.

**Q.11. Why are conformance levels important?**

Conformance levels are important because they allow products having different levels of capability to participate in the ODBC standard. The standard need not conform to the lowest level, nor need it address only the capabilities of the highest level products.

**Q.12. Explain the difference among the three types of data sources.**

User pertains to a single user on a single machine. File holds the data source data in a file that can be shared among users – perhaps on different machines. System pertains to a particular computer.

**Q.13. What are two tasks to be accomplished when setting up an ODBC data source name?**

1. Pick the DBMS driver 2. Pick a particular database.

---

## Multiple Choices

1. **Which can only interact with relational database and table-like data structures?**
   a. OLE DB b. ODBC c. ASP d. All
2. **Which of the following is true about ODBC?**
   a. ODBC has experienced little practical success.
   b. ODBC requires developers to have a thorough knowledge of many DBMS native libraries.
   c. ODBC can be used to access data from spreadsheets.
   d. ODBC has an object-oriented interface.
3. **According to ODBC standard, which of the following is NOT part of specification of a data source?**
   a. Associated DBMS b. Database c. The driver d. Operating system
4. **The ODBC standard defines a means of doing which of the following?**
   a. Start transactions b. Rollback transactions c. Create a connection d. All
5. **Which of the following is a function performed by the driver manager in ODBC?**
   a. Submit SQL statements to the data source
   b. Determine the type of DBMS that processes a given ODBC data source
   c. Load the appropriate ODBC driver into memory
   d. Both b and c
6. **The intermediary between application and DBMS drivers in ODBC architecture is:**
   a. Driver manager b. OLE DB interface c. ODBC driver d. Data source
7. **Which of the following is a task performed by the driver according to the ODBC standard?**
   a. Determines the appropriate DBMS
   b. Validates the format of the ODBC command received from the application
   c. Converts data source error codes into ODBC standard error codes
   d. Verifies the application to the data source
8. **The _____ processes ODBC requests and submits specific SQL statements to a given type of data source.**
   a. Driver manager b. ADO c. Driver d. Source converter
9. **A data source that is fully SQL-compliant would use what type of DBMS driver?**
   a. Single-tier b. Multiple-tier c. SQL transform d. Text-based

**10. How does application determine the level of ODBC conformance available from a driver?**
a. Application makes a call to driver manager
b. The application makes a call to the data source.
c. The application makes a call to the driver.
d. The developer must determine the level of conformance before the application is written.

**11. A _____ data source can be shared among database users as long as they have the same DBMS driver and privilege to access the database.**
a. File b. Common c. Shared d. System

**12. ODBC interfaces are abstractions of_____**
a. OLE objects b. Native DBMS access methods
c. Driver managers d. DBMS data sources

**13. A driver _____ is a set of functions that the application can call to receive services**
a. OLE Object b. DBMS data Source c. API d. System

**14. _____ Levels specify which SQL statements, expressions and data types a driver can process**
a. SQL conformance b. API c. ODBC conformance d. None

## Answers

| | | | |
|---|---|---|---|
| 1. b | 2. c | 3. c | 4. d |
| 5. e | 6. a | 7. c | 8. c |
| 9. a | 10. c | 11. a | 12. b |
| 13. c | 14. a | | |

## True/False

1. ODBC is the foundation of the Microsoft data access world
2. An application that uses the ODBC interface could process any database that is ODBC-compliant without any program coding changes.
3. ODBC identifies two types of drivers: single tier and multiple tier.
4. ODBC has not had practical success, but has shown great potential for future development.
5. ODBC works with table-like data sources such as relational databases and spreadsheets.
6. ODBC is an object-oriented interface to access ADO objects.
7. OLE DB can be used to access ODBC data sources.
8. With ODBC, driver manager serves as intermediary between application and DBMS drivers.
9. With ODBC, only single driver is needed to handle all data source types like Oracle DB2 etc.
10. In ODBC, the amount of work that the driver must do is largely determined by the degree of SQL-compliance of the data source.
11. A multiple-tier ODBC driver may reformat an SQL request, but it does not actually process the SQL statement.
12. With ODBC, a database and the DBMS that processes it are identified by the data source.
13. With ODBC, a file data source is available only to the user that created it.
14. In general, best type of ODBC data source to define for Web application is system data source.
15. An API is a set of functions that perform a particular task
16. DBMS driver receives processing requests from application program and sends to DBMS

## Answers

| | | | |
|---|---|---|---|
| 1. T | 2. T | 3. T | 4. F |
| 5. T | 6. F | 7. T | 8. T |
| 9. F | 10. T | 11. T | 12. T |
| 13. F | 14. T | 15. T | 16. T |

CHAPTER 11

# Distributed & Object Oriented Databases

## Chapter Overview

## 11.1 Centralized Database System

A type of database system in which all system components reside at a single computer or site is known as **centralized database system.** The system components include database and DBMS etc. The users access the centralized database system remotely via terminals connected to the site. However, all data access and processing takes place at the central site.

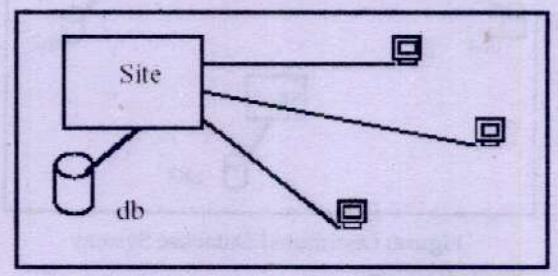



**Figure:** Centralized database system

## 11.2 Distributed Database System

A type of database system in which the database is stored physically across computers or sites at different locations is known as distributed database system. The computers in distributed database system are connected together by data communication network.

The sites of a distributed database may be spread over a large area connected via a **wide area network (WAN)** or over a small area such as a building or a campus connected via local area network (LAN). The computers may be of different types, managed by different operating systems and each fragment of the database may be managed by a different DBMS.

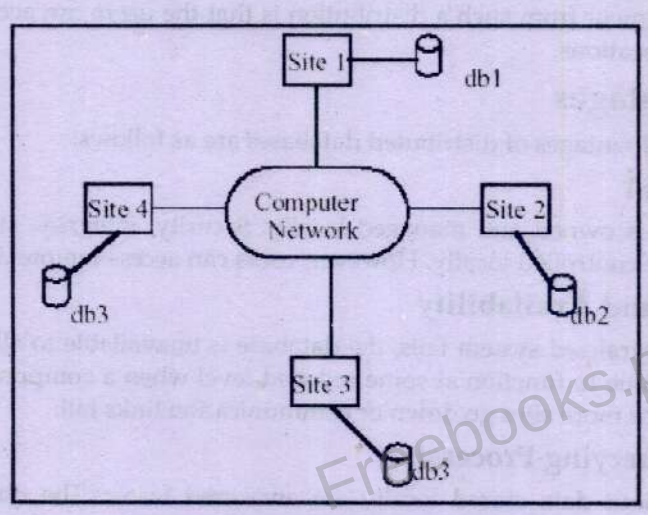



**Figure:** Distributed Database System

## 11.3 Decentralized Database

A type of database system in which the database is stored on computers or sites at different locations is known as decentralized database system. In a decentralized database, the computers are not interconnected via a network. So the users at various sites cannot share data. It is a collection of independent databases.

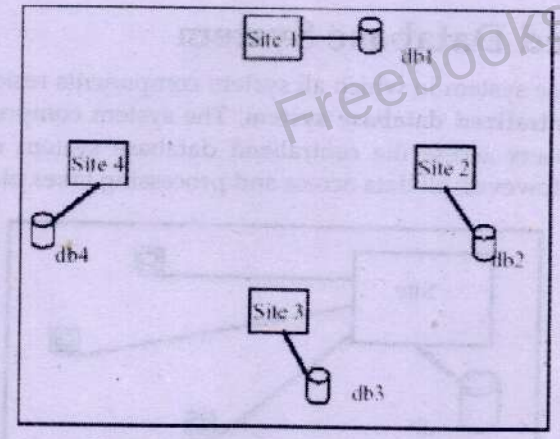



**Figure:** Distributed Database System

## 11.4 Distributed DBMS

The users access the distributed database via applications. In a distributed database system database, the applications running at any of the system's sites should be able to operate on any of the database fragments. The software that manages a distributed database in such a way is called a **distributed database management system (DDBMS)**.

### 11.4.1 DDBMS Motivation

Many organizations are naturally distributed over different locations. For example, a company may have locations at different cities, or a bank may have multiple branches. It is natural for databases used in those organizations to be distributed over these locations. An important requirement from such a distribution is that the users can access data both locally and at the other locations.

### 11.4.2 Advantages

Different advantages of distributed databases are as follows:

#### 1. Local Control

Local data is owned and managed locally. Security, integrity, storage representation and hardware are controlled locally. However, users can access remote data when necessary.

#### 2. Reliability and Availability

When a centralized system fails, the database is unavailable to all users. A distributed system will continue to function at some reduced level when a component fails. It continues to operate if one or more sites go down or communication links fail.

#### 3. Efficient Querying Processing

Queries about data stored locally are answered faster. The queries can be split to execute in parallel at different sites or they can be redirected to less busy sites.

#### 4. Modular Growth

It is much easier to add another site in distributed system than to expand a centralized system. Suppose that an organization expands to a new location or adds a new work group. It is often easier and more economical to add a local computer and its associated data to the distributed network, than to expand a large central computer.

### 5. Economics

It costs much less to create and maintain a system of smaller computers with the equivalent power of a single large computer.

## 11.4.3 Disadvantages

Some disadvantages of distributed database system are as follows:

### 1. Software Complexity & High Costs

A DDBMS is more complex than a centralized DBMS. Therefore, it is more expensive to buy and maintain. There are also additional manpower costs to manage and maintain the local DBMSs and the underlying network.

### 2. Processing Overheads

The various sites must exchange messages and perform additional calculations to ensure proper coordination among data at the different sites.

### 3. Data Integrity

It is harder to enforce data integrity when data is updated at different sites simultaneously.

### 4. Complex Database Design

Besides the normal difficulties of designing a centralized database, the design of a distributed database has to take account of fragmentation and replication of data and allocation of fragments.

### 5. Slow Response

If the data are not distributed properly according to their usage, or if queries are not formulated correctly, response to requests for data can be extremely slow.

# 11.5 Distributed Database Design

The methodology used for the logical design of a centralized database applies to the design of a distributed one as well. However, for a distributed database three additional factors have to be considered:

## 11.5.1 Data Fragmentation

The database may be broken up into logical units called fragments. These fragments are stored at different sites. Data fragmentation is the design process of deciding what the fragments will be. The simplest logical units are the relations themselves. Each relation is stored at a particular site. In many cases, it makes sense to divide a relation to smaller logical units for distribution.

### 11.5.1.1 Horizontal & Vertical Fragmentation

A vertical partition, or vertical fragment, refers to a table that is broken into two or more sets of columns. A horizontal partition, or horizontal fragment, refers to the rows of a table when they are divided into pieces.

## 11.5.2 Data Replication

A copy of each fragment may be stored at several sites. Data replication is the design process of deciding which fragments will be replicated.

## 11.5.3 Data Allocation

Each fragment has to be allocated to one or more sites, where it will be stored. There are three strategies regarding the allocation of the data:

### 11.5.3.1 Fragmented Data Allocation

The database is partitioned into disjoint fragments. Each fragment is assigned to one site and there is no replication. This is also called **non-redundant allocation.**

#### Advantages

Some advantages of this data allocation method are as follows:

- **Efficiency:** Data are stored close to where they are used and separate from non-local data. So it can be accessed and used efficiently.
- **Security:** Data that is not relevant at a particular site are not made available. It provides more security.

#### Disadvantages

Some disadvantages of this data allocation method are as follows:

- **Inconsistent access speeds:** Access to remote fragments takes much longer than to the local one.
- **Back up Vulnerability:** Since data are not replicated, problems at a site make its fragment inaccessible.

### 11.5.3.2 Complete Replication

A complete copy of the database is maintained at each site and there is no fragmentation.

#### Advantages

Some advantages of this data allocation method are as follows:

- **Faster & Consistent Access Speeds:** Each site that has a full copy can process queries locally, so queries can be processed rapidly
- **Reliability:** If one of the sites containing a relation (or the database) fails, a copy can always be found at another site. The available copies can all be updated as soon as possible as transactions occur. The unavailable nodes can be updated.
- **Node Decoupling:** Each transaction may proceed without coordination across the network. So, if nodes are down, busy, or disconnected, a transaction is handled when the user desires.

#### Disadvantages

Some disadvantages of this data allocation method are as follows:

- **Update complexity:** Whenever there is an update to one of the database copies, the update has to be applied to all the other copies.
- **Increased Storage Requirements:** Each site that has a full copy must have the same storage capacity that would be required if data were stored centrally. Each copy requires storage space and processing time is required to update each copy on each node.

### 11.5.3.3 Selective Replication

Selective replication is a combination of fragmentation and replication. The objective of this strategy is to have the advantages of the previous approaches without the disadvantages.

## 11.6 Functions of a DDBMS

A distributed DBMS provides access to data at the various sites. A DDBMS is required to perform the following functions in addition to the functions of a DBMS:

- **Extended communication services** to provide access to remote sites and allow the transfer of queries and data among them.
- **Extended system catalog** (Global System Catalog) to store data distribution details; this way can determine the location from which to retrieve requested data.
- **Distributed query processing**. If necessary, translate the request at one site using a local DBMS into the proper request to another site using a different DBMS and data model.
- **Extended concurrency control** to maintain consistency of replicated data.
- **Extended recovery services** to be able to recover from individual site crashes and failure of communication links.

## 11.7 Client/Server Architecture

The client-server architecture has been developed to deal with new computing environments in which a large number of computers, workstations, file servers, printers, and other equipment are connected together via a network. The idea is to define specialized servers with specific resources and functionalities, which many clients can access and use.

The client-server architecture is increasingly being incorporated into commercial DBMS packages. The DDBMS software is divided into two levels i.e. **client** and **server** to reduce its complexity. Some sites may run the client software only other sites may be dedicated server machines that run the server software only. Some sites may support both client and server modules.

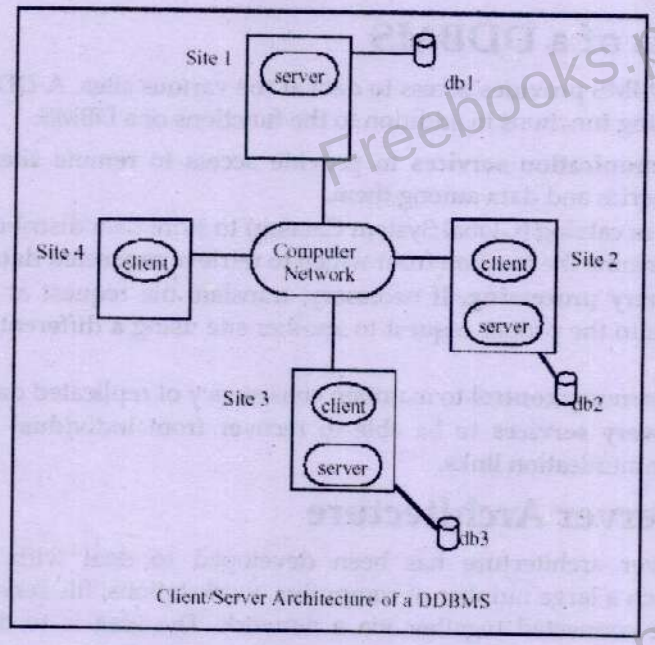
Different approaches have been proposed to divide the DDBMS functionality between client and server. However, a number of relational DDBMS's have taken the client-server SQL approach that:

- The server and client software, communicate with each other using SQL.
- The server software, also called **back-end** machine, is responsible for local data management at a site, much like centralized DBMS software.
- The client software, also called **front-end** machine, is responsible for most of the distribution functions, user interface and programming language interface functions.

Another function controlled by the client is to ensure consistency of replicated copies of data by distributed concurrency control techniques. It must guarantee the atomicity of global transactions through global recovery when certain sites fail.

Interaction between client and server might proceed as follows during the processing of an SQL query:

- The client parses a user query and decomposes it into a number of independent site queries. Each site query is sent to the appropriate server site.
- Each server processes local query and sends the resulting relation to the client site.
- The client site combines the results of the subqueries to produce the result of the originally submitted query.



Client/Server Architecture of a DDBMS

One important function of the client is to hide the details of data distribution from the user. This property is called **distribution** or **location transparency**. If this is supported the user can write queries and transactions as if the database is centralized. If the DDBMS does not offer distribution transparency, the users has to specify the sites at which the data referenced in queries reside.

The client-server architecture we have discussed above has two tiers. However, new **three-tier architecture** for client-server database applications has emerged. This architecture has a **data management layer**, an **application layer** and a **user interface layer**. The data management layer holds the database schema and data. The application layer holds the programs that embody the application logic. The user interface layer manages the forms and reports that are presented to the user. The three-tier architecture complements client-server computing. The application layer is a client with regard to the data management, and the user interface layer is a client with regard to the application layer.

## 11.8 Types of DDBMS

A DDBMS can be classified as follows:

### 11.8.1 Homogeneous

The term homogeneous database means that the database technology is the same or at least compatible at each of the locations. The data at the various locations are also compatible. Homogeneous databases simplify the sharing of data among the various users.

#### Characteristics of Homogeneous DDBMS

- Data are distributed across all the nodes.
- The same DBMS is used at each location.
- All data are managed by the distributed DBMS so there is no exclusively local data.
- All users access the database through one exclusive schema or database definition.
- The global schema is simply the union of all the local database schemas.

### 11.8.2 Heterogeneous Databases

In heterogeneous distributed database potentially different DBMSs are used at each node. It is difficult in most organizations to force a homogeneous environment, yet heterogeneous environments are much more difficult to manage.

#### Characteristics of Heterogeneous DDBMS

- Data are distributed across all the nodes.
- Different DBMS is used at each location.
- Some users require only local access to databases, which can be accomplished using only the local DBMS and schema.
- A global schema exists, which allows local users to access remote data.

## 11.9 DBMS Transparency & Gateways

The term **DBMS transparency** refers to the ability to hide the knowledge that the local DBMSs may be different. It applies to a heterogeneous environment. It is one of the most difficult transparencies to provide.

To achieve this transparency, DBMSs at different sites should support the same interface. Thus, for example an ORACLE and a SYBASE DBMS both of which support the official SQL might be able to behave as equal partners in a heterogeneous distributed database system.

Suppose the existence of a distributed heterogeneous database system that resides at site X on a machine running SYBASE and at site Y on a machine running ORACLE. A SYBASE user at X site wants to see a single distributed database that includes data from both sides. SYBASE must provide an application program that usually referred to as a **gateway** that runs on top of ORACLE and which has the effect of making ORACLE look like SYBASE.

A gateway from system X to system Y must provide the following functions:

- Protocols for the exchange of information between two DBMSs.
- Provide a relational server function for the Y DBMS, i.e. execute arbitrary unplanned SQL commands on the Y DBMS.
- Mapping between the two DBMSs data types
- Mapping the SQL of the X system to that of Y
- Mapping feedback information of the y system to that of the X
- Mapping the Y system's catalogue to that of the X
- An effective mechanism to synchronize locking & commit mechanisms of the Y system with those requested by the X system.

## 11.10 Object Oriented Databases

Object oriented programming (OOP) methodology has altered the way we program. Object oriented ideas are also influencing how we think about other concepts such databases. The merging of object oriented programming and database management systems is the direct result of the increasingly difficult task of dealing with complex data requirements and the issues of modeling real world data with relational database technology.

The issues such as the increasing use of multimedia in databases, the size of databases, and the complexity of today's databases, require solutions far beyond relational databases. Object Oriented Database Management Systems (OODBMS) are a direct result of the emerging database technologies from the reorganization of information systems.

## 11.11 Object Oriented Data Model

To fully understand the OODB and its difference with relational models, we must understand some terms of OODM. At the highest level of abstraction, an object is modeling a real-world situation or entity. It has unique identification, properties, and the ability to work in conjunction with other objects. An entity in an ER model does not provide this behavior.

**Object Identity (OID)** provides the unique identity of an object. An OID is assigned by the system and it is unable to be changed. Therefore, no two objects created can have the same OID. Removing an object from the system removes an OID from the system and by definition the system will never assign another object the removed object's OID.

An object is defined by the **attributes** that describe the real-world entity. In object-oriented terminology, these attributes as **instance variables**. Attributes can be of any type supported by a programming language. In addition to data attributes, objects use functional attributes or **methods** that describe the functionality of an object. Methods are used to access data within the object. Some methods also access the attributes of other objects.

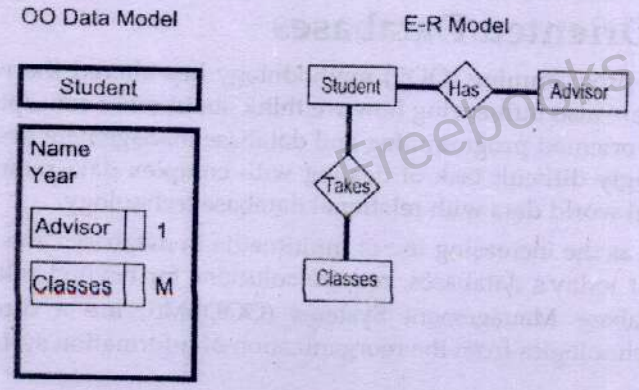
### 11.11.1 Characteristics of the OODM

OODM must support following characteristics also:

- The OODM must be able to provide complex object representation.
- The OODM must be extensible. It means it must provide support for defining new data types and methods that are capable of operating on that object.
- The OODM must support encapsulation or information hiding. It must be able to hide how the data is represented within the object and how each method is implemented from other objects and other entities outside of itself.
- The OODM must exhibit inheritance. Objects will exist in a hierarchy relationship. A hierarchy relationship is a relationship where an object inherits from a root object. This provides the ability to take on the attributes (data) and operations of other objects above it in the hierarchy.
- The OODM must support OIDs.

## 11.12 Object Oriented Database

The following figure compares the OODM with the ER model. The two models depict a Student who has an Advisor and takes Classes. The OODM is a class containing the attributes Name and Year. It then serves as a container class that contains an Advisor object/entity and a Classes object/entity.



On the other hand, the ER model has a three entity tables: Student, Advisor, and Classes. In addition, the ER model must have two relationship tables to join the information in a meaningful manner. As one can see, the ER technique needs two more entities in order to provide the relationships that the OODM approach inherently suggests.

### 11.12.1 Entity Sets and Tables

OODM's idea of a class resembles the ER model's idea of an entity set or table. Like an object, OODM classes are more powerful then ER model idea of an entity set or table. A class not only describes data structure but also describes the behavior of the class objects. Features such as methods, which allow the description of behavior of objects, give OODB full Abstract Data Type (ADT) capabilities allowing an increase in the semantics of the object being modeled. This full ADT capability also allows for encapsulation and inheritance support.

### 11.12.2 Relationships

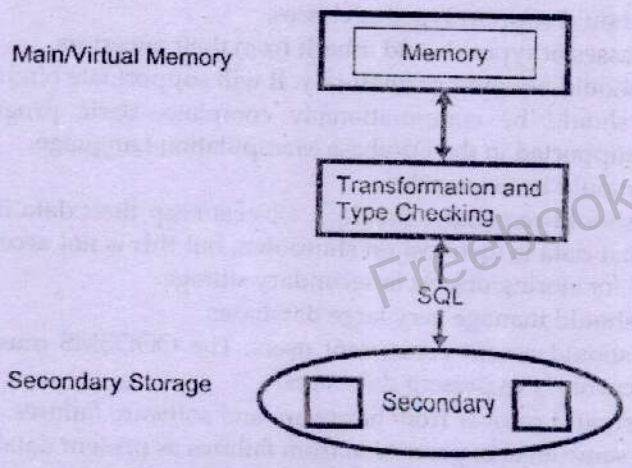
An important property of a model is how it represents relationships among different components of data. OODB has two types: inter-class or class hierarchy. This is contrasted against the ER value-based relationships. Value-based relationships are established on equal planes or equal data types between two tables. In sharp contrast, relationships in the OODB are OID based. This allows independence between the object's state and the relationship.

### 11.12.3 Versioning

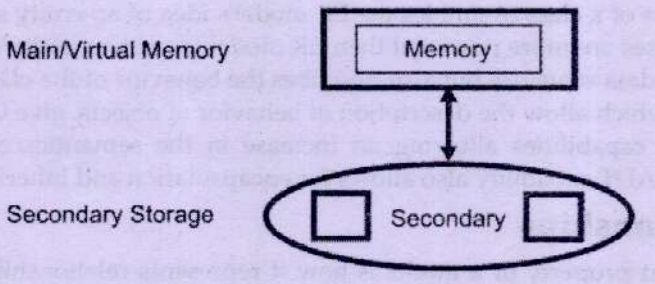
Another difference between traditional and E-R based databases is the OODBs support of versioning. Just as a document can have multiple versions so can a database. One is able to load a system and change aspects to determine how this would affect the overall system. Then the original system can be reloaded intact.

### 11.12.4 Data Access

Traditionally, data access can be viewed as a two level storage model. It needs a link between main memory and secondary storage device to access data contained in database. The user needs data to analyze and use it. In order to get the data, the user makes a query through SQL. SQL accesses the data on a secondary storage medium. It then transforms and type checks the data before it transfers the data to the user as shown in the following figure:



The OODB eliminates the two-level storage view and provides a single-level view. An OODB does not need SQL or a transformation/type checking phase in order to bring the data from secondary storage into memory. The user instantiates an object. This object is the link from memory to the information on the secondary storage. The user then interacts with the object, which is in memory to receive data from secondary storage.




Another aspect of data access is the issue of **late** versus **early binding**. In a traditional ER model, data type is bound early or at the definition of the table structure. In contrast, an OODB allows for late binding of data types or binding of data types at run-time. In such an environment, an object's instance variable's type is not known until it is actually utilized, thus allowing any type of data to be assigned.

## 11.12.5 OODM + DBMS = OODBMS

An OODBMS actually manages the data access and the querying of the data from the databases. It manages multiple users trying to access the data at the same time. It provides recovery services and manages all the databases in its system. An OODBMS is an important component, since it provides and manages the constraints for the database.

The "Object-Oriented Database System Manifesto" was introduced at the First International Conference in Deductive and Object-Oriented Databases in Kyoto, Japan. It provided the following musts for an OODBMS:

- OODBMSs should support complex objects.
- OODBMSs should support object identity.
- OODBMSs should encapsulate objects.
- OODBMSs should support types or classes.
- OODBMS classes or types should inherit from their ancestors.
- OODBMSs should not bind prematurely. It will support late binding.
- OODBMSs should be computationally complete. Basic programming language notions are supported in the Database Manipulation Language.
- OODBMSs should be extensible.
- OODBMSs should remember the data. Objects keep their data in memory, unlike a database. That data is then lost on shutdown but this is not acceptable. A way must be provided for storing objects in secondary storage.
- OODBMSs should manage very large databases.
- OODBMSs should accept concurrent users. The OODBMS must support the same level of concurrency as present databases.
- OODBMSs should recover from hardware and software failures. The OODBMS must support the same level of protection from failures as present databases.
- OODBMSs should have a simple way of querying data. A method for efficient querying must be available, similar to SQL.

The Manifesto is the first attempt at describing a standard on which OODBMSs should be based. It is an important first step toward an agreement on the minimum an OODBMS should support. In addition, since it is the most significant list of requirements to be assembled, most OODBMSs are measured against it.

### 11.12.6 Advantages of OODBMS

OODBMSs provide many advantages. These advantages are important because they solve many of the problems traditional systems cannot solve.

- First, the amount of information that can be modeled by an OODBMS is increased, and it is also easier to model this information.
- OODBMSs provide complex objects that allow ease of integration for multimedia, CAD, and other such specialized databases.
- OODBMSs are also able to have higher modeling capabilities through extensibility.
- With an OODBMS, one would be able to add more modeling capabilities, thus being able to model even more complex systems. This extensibility provides a solution for incorporating future and existing databases into one environment.
- In an OODBMS, versioning is available to help model various changes to systems. With versioning, one would be able to revert to previous data sets, and compare the current sets to the previous.
- Reusability of classes plays a vital role in faster application development and maintenance. Generic classes are powerful, but more importantly they can be reused. Since classes can be reused, redundant material does not need to be designed. This leads to faster production of applications and easier maintenance of those applications and databases.

### 11.12.7 Disadvantages of OODBMS

In addition to modeling advantages, OODBMSs have system advantages as well.

- Traditional ER systems have been used for a long time and a change would stray from established ideas. It would require people to think differently, and in some cases relational users lack the OO foundations needed to work with OODBMSs.
- Educating people on the OO foundations is a very difficult process. It would require a significant amount of time, money, and other resources.
- Because of the change even more time would be required to move the data into the new OODBMSs.
- Traditional systems and OODBMS must understand each other and the relations they are representing. Again, this would take time, money, and resources.
- There is not query language such as SQL. While it is easier to make complex queries with OODBMSs, no query language exists.
- Furthermore, there are no standards for design and implementation in place. OODBMS can solve problems of traditional systems, but a standard is required.

## 11.13 Future Developments

Future developments for the OODB must include an ad hoc querying language for the average user. This language should provide for OODBs what SQL provides for traditional databases. Also, a standard for design, notation, and implementation must be agreed upon. Future developments for the OODB could include an easier accessing method from the Internet and integration of ideas such as XML.

## Short Questions

**Q.1. Contrast the following terms:**

Distributed and Decentralized database

Location Transparency and local Autonomy

**Distributed database and decentralized database:** Both distributed and decentralized databases are stored on computers at multiple locations. In decentralized database, network does not interconnect the computers, so that users at various sites cannot share data. Thus it is best regarded as a collection of independent databases, rather than having the geographical distribution of a single database.

**Location transparency and local autonomy:** In a distributed database, the network must allow users to share the data as transparently as possible (location transparency), yet must allow each node to operate autonomously (local autonomy) when network linkages are broken or specific nodes fail.

**Q.2. Discuss five advantages of distributed database system as compared to centralized system.**

**1. Increased availability and reliability:** When centralized system fails, database is unavailable to all users. A distributed system continues to function at some reduced level when a component fails.

**2. Local control:** Distributing data encourages local groups to exercise greater control over their data that promotes improved data integrity and administration.

**3. Modular growth:** It is often easier and more economical to add a local computer and its associated data to the distributed network than to expand a large central computer.

**4. Lower communication costs.** With a distributed database data can be located closer to point of use. This can reduce communication costs, compared to a central system.

**5. Faster response:** Most requests for data by the local users can be satisfied by data stored at the local site. It speeds up query processing as communication and central computer delays are minimized.

**Q.3. What is the difference between a homogeneous and heterogeneous DDBMS? Under what circumstances would such systems generally arise?**

In a homogeneous system, all sites use the same DBMS product. In a heterogeneous system, sites may run different DBMS products, which need not be based on the same underlying data model, and so the system may be composed of relational, network, hierarchical, and object-oriented DBMSs.

Homogeneous systems are much easier to design and manage. This approach provides incremental growth, making the addition of a new site to the DDBMS easy, and allows increased performance by exploiting the parallel processing capability of multiple sites. Heterogeneous systems usually result when individual sites have implemented their own databases and integration is considered at a later stage.

**Q.4. What functionality do you expect in DDBMS?**

A Distributed DBMS co-ordinates the access to data at the various sites the distributed database resides. DDBMS is required to perform the following functions in addition to the functions of a DBMS:

- Extended communication services to provide access to remote sites and allow the transfer of queries and data among them.
- Extended system catalog (Global System Catalog) to store data distribution details; this way can determine the location from which to retrieve requested data.
- Distributed query processing. If necessary, translate the request at one site using a local DBMS into the proper request to another site using a different DBMS and data model.
- Extended concurrency control to maintain consistency of replicated data.
- Extended recovery services to be able to recover from individual site crashes and failure of communication links.

**Q.5. Define the terms partitioned and replicated as they pertain to distributed database applications.**

Partitioned data refers to sections or pieces of the database that are distributed on different computers. Replicated refers to whether or not data is duplicated on more than one computer.

**Q.6. Explain the difference between a vertical fragment and a horizontal fragment.**

A vertical partition, or vertical fragment, refers to a table that is broken into two or more sets of columns. A horizontal partition, or horizontal fragment, refers to the rows of a table when they are divided into pieces.

**Q.7. What is object identity (OID)?**

Object Identity (OID) provides the unique identity of an object. An OID is assigned by the system and it is unable to be changed. Therefore, no two objects created can have the same OID. Removing an object from the system removes an OID from the system and by definition the system will never assign another object the removed object's OID.

## Multiple Choices

1. **Which is NOT a characteristic of a homogeneous distributed database environment?**
   a. The same DBMS is used at each location.
   b. All users access the database through one global schema.
   c. All data are managed by the distributed DBMS
   d. Some users require only local access to data.
2. **Which is NOT a characteristic of a heterogeneous distributed database environment?**
   a. Different DBMSs may be used at each location.
   b. Data are distributed across all the nodes.
   c. Global schema exists, which allows local users to access remote data.
   d. All data are managed by the distributed DBMS
3. **Which is NOT an advantage to using horizontal partitioning for a distributed database?**
   a. Security b. Global optimization c. Efficiency d. Ease of querying
4. **Advantages to data replication include**
   a. Fast response and storage requirements b. Reliability and fast response.
   c. Complexity of updating and node decoupling d. Cost of updating and fast response
5. **A vertical fragment or partition refers to**
   a. Table that is broken into two or more sets of rows
   b. Table that is broken into two or more sets of columns
   c. Database downloaded in a file-sharing system
   d. The data measures held constant in a data cube
6. **A horizontal fragment or partition refers to**
   a. a table that is broken into two or more sets of rows
   b. a table that is broken into two or more sets of columns
   c. a database downloaded in a file-sharing system
   d. the data measures held constant in a data cube
7. **The greatest disadvantage of distributed databases is the availability**
   a. Security risk b. Cost/complexity
   c. Difficulty of control and possible integrity problems d. None

## Answers

| | | | |
|---|---|---|---|
| 1. d | 2. d | 3. b | 4. b |
| 5. b | 6. a | 7. c | |

CHAPTER 12

# Introduction to MS Access

## Chapter Overview

# 12.1 Microsoft Access

Microsoft Access is a **relational database management system (RDBMS)**. It is used to store and manipulate a large amount of information. It is very easy to understand for users. Its graphical interface helps the users to create queries, forms and reports easily. Even an inexperienced programmer can use MS Access to perform different activities. The process of entering, updating and reporting information becomes very easy.

# 12.2 Starting MS Access

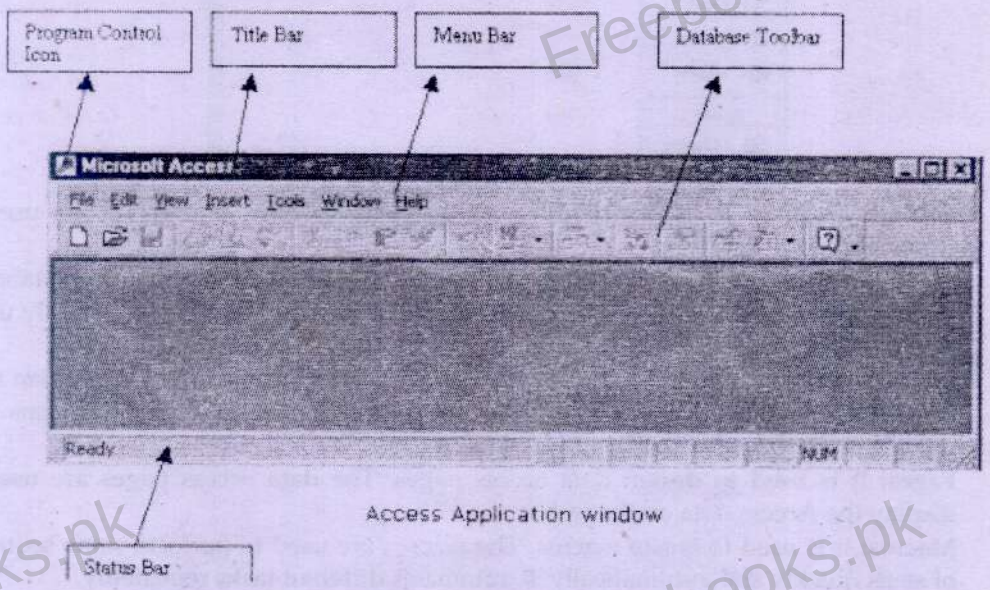
The following procedure can be used to start MS Access:

1. Click **Start** button Start on **Taskbar** and select **Programs** menu item. The **Programs** submenu will be displayed.
2. Click **Microsoft Access** program item. MS Access will be started. **OR**

- Double click on **Microsoft Access** icon on Desktop if it appears.

## 12.2.1 Application Window of MS Access

MS Access Application window follows standard layout of all Microsoft applications. It contains different object that are used to design and create databases.



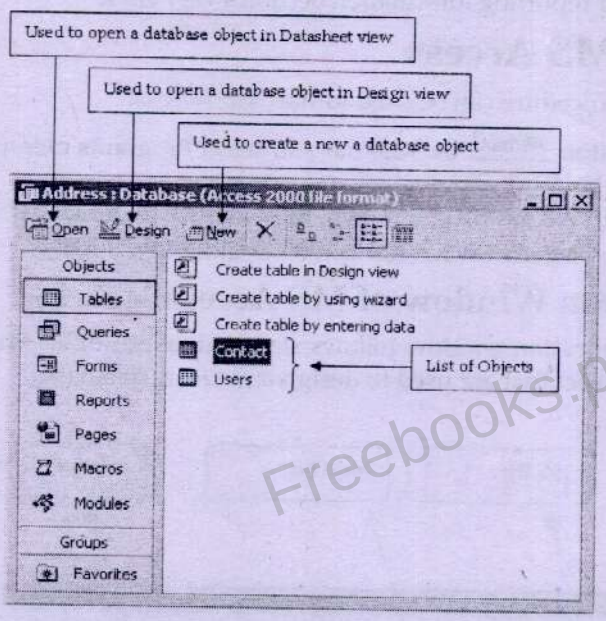

Access Application window

Some important parts of application window are as follows:

- **Title Bar:** Title bar identifies the application that is running (Microsoft Access)
- **Menu Bar:** Menu bar contains different menus that used to issue various commands.
- **Toolbars:** Toolbars contain icons that are shortcuts to select commands from a menu.
- **Menus:** Menus contains different menu options to execute various commands.
- **Scroll Bars:** Scroll bars are used to move around the window if its contents do not fit on screen.
- **Status Bar:** Status bar displays information while the user is working on an object.

## 12.2.2 Database Window in MS Access

The **database window** in MS Access is used to organize all objects in the database. It contains its own title bar and toolbar. Database window is divided into two parts. The left side of database window contains seven buttons. Each button indicates different type of object used to develop database application. The right side of database window displays the list of different objects. Different buttons in the database window are as follows:




- **Tables:** It is used to create, modify and manipulate tables. The tables are used to store data in the database.
- **Queries:** It is used to create queries. Queries are used to retrieve data from database.
- **Forms:** It is used to create forms. Forms are used to enter data in tables easily using graphical user interface. The form consists of buttons, textboxes and lists etc.
- **Reports:** It is used to create reports. Reports are used to display the information from database in different ways. The reports are used to make important decisions. The reports can be based on tables or queries.
- **Pages:** It is used to design data access pages. The data access pages are used to display the Access data on the web.
- **Macros:** It is used to create macros. The macros are used to perform same sequence of steps quickly and automatically. It automates different tasks repeatedly.
- **Modules:** It is used to create modules. A module contains an object that stores the code of VBA **(Visual Basic for Applications)**.

# 12.3 Creating and Closing a database

The following procedure is used to create a database:

1. Click **File > New...** MS Access will display the Task Pane.
2. Select **Blank Database** option. A dialog box will appear.
3. Select the desired location.
4. Enter any file name.
5. Click **Create** button. A blank Database window will appear.

If the user chooses to create a new database using Database Wizard, Access prompts the user to choose the required template required. The user then gives the database name and selects the relevant options from **Database Wizard Window**. These options vary depending on the selected template.

### Creating New Database using Templates

The following procedure is used to create a database using templates:

1. Click **File > New...** MS Access will display the Task Pane.
2. Select **General Template** option. A dialog box will appear.
3. Select the desired location.
4. Enter the desired file name.
5. Answer the wizard's questions.
6. Select **Next** and then select **Finish**.


| Choices | Description |
|---|---|
| Open a File | It allows the user to access an existing database quickly. It also displays the last four databases used. |
| New | It creates a blank database allowing the user to add tables, forms, reports and other objects later. |
| New from File | It creates a new file copying the structure and data contents of an existing Access file. |
| New Template | It creates the required database elements in one operation such as tables, forms and reports |

### Closing a Database

- Click **Close** icon on **Database** Window title bar **OR**
- Select the **Database** Window from **File** menu and select **Close**.

## 12.4 Exiting Access

A user can exit from **Access** in many ways. If there is any unsaved object, MS Access prompts to save it before closing. The following methods are used to exit MS Access:

- Double click Access Program Icon at the top-left of Access window **OR**
- Click **Close** button at the top-right of Access window

## 12.5 Creating table in MS Access

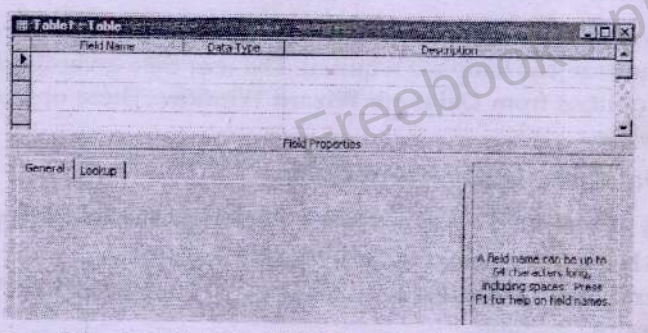
A table is created in a database. The option **Create table in Design view** is the most common way of creating a table. Different methods of creating a table are as follows:

- Creating table in Design view
- Creating table by wizard
- Creating table by entering

### 12.5.1 Creating Table in Design View

The **Design View** is used to define the fields of a table. The window is divided into two parts. The following procedure is used to create a table in design view:

- Create new blank database.
- Double click **Create table in Design view.** The **Design View** will appear as follows:




- Type the name of the first field in **Field Name** column.
- Press tab key to move to **Data Type** column and select required data type for field.
- Press **Tab** key to move to **Description** column. It is used to enter comments about the field. This is optional.
- Press **Tab** key to move to **Field Name** for the next field.
- Repeat the above steps for entering any number of fields in the table.

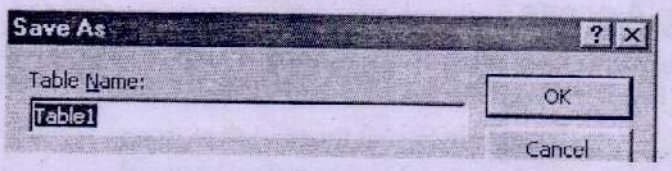
### Setting Primary key

- Select the field for primary key.
- On the **Table Design** toolbar, click **Primary Key** OR
- From the **Edit** menu, select **Primary Key**. The row that is chosen as the primary key is marked with a small symbol in the selector button.

### Saving a Table

The following steps are performed to save a table.

- On the **Table Design** toolbar, click **Save** OR
- From **File** menu, select **Save**. The **Save As** dialogue box will appear.




- Enter a table name and click **OK**. The table will be saved. The new table will appear in the main database as follows:

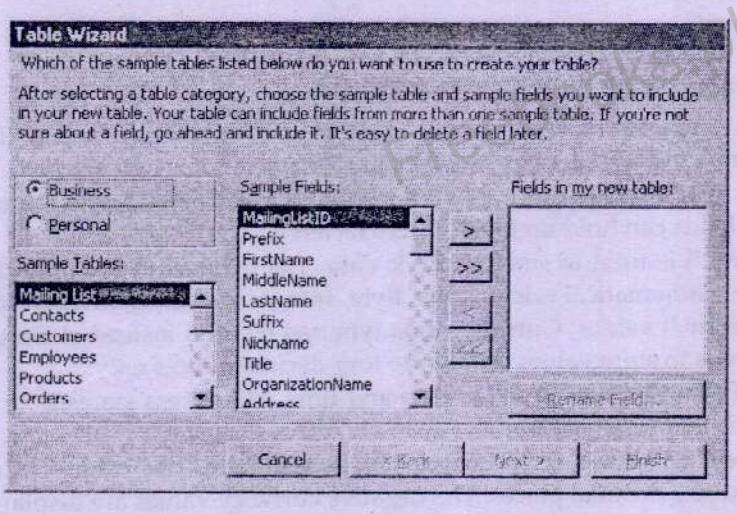
## 12.5.2 Create a Table using Table Wizard

MS Access table wizard provides an easy way to create tables. It provides various table templates to create business and personal tables. The wizard helps the user to create common types of tables to manage mailing lists, recipes, investments and video collection etc.

### Create a Table using Table Wizard

The following procedure is used to create a table using table wizard:

1. Open Database window.
2. Click **Tables** Tables in **Objects** bar.
3. Double click **Create table by using wizard**. The **Table Wizard** will start.




The sample tables are grouped in two categories **Business** and **Personal**.

4. Select appropriate category and type of table to be created.
5. Select appropriate fields for the table. The four video controls are used to perform the following operations:

| Button | Description |
|---|---|
| > | It moves the highlighted field into **Fields in my new table** list box. |
| >> | It moves all fields into the Fields in my new table list box. |
| < | It moves a selected field into the Sample Fields list box. |
| << | It moves all fields into the Sample Fields list box. |

6. Select the field to rename.
7. Click on **Rename Field** button. The **Rename Field** dialog box will appear as follows:
8. Enter new name of the field and click **OK**.
9. Select the appropriate fields and click **Next >**.
10. The default name of table is the name of first sample table. It can be changed.
11. The wizard also asks to select a primary key for the table.
12. Click **Next >**.
13. The final screen of wizard offers the choice to modify table design, enter data in table in datasheet view or create a form to use to enter data.
14. The user can also choose to display an appropriate help topic that gives advice on entering and modifying data.
15. Select any options.
16. Click **Finish**.

## 12.5.3 Creating Table by Entering Data

MS Access provides the facility to create table by directly entering data. This option provides a blank datasheet. The user can enter data in cells and click **Save** button. It will prompt to add a primary key field. The fields are given names such as **Field1**, **Field2** etc. The names can be modified later by the user. The data types of the fields are automatically specified according to the data enter in table.

## 12.6 Data Types

MS Access provides the following data types:

- **Text:** It is used to store alphabetic, numbers and special characters. It can store up to 255 characters. The default length is 50 characters if it is not specified.
- **Memo:** It is used to store lengthy text. It is normally used to store comments etc. Memo fields can hold up to 64,000 characters.
- **Number:** It is used to store numeric data. The fields with **Number** data type can be used in mathematical calculations. **Byte**, **Integer** and **Long Integer** data types cannot store decimal values. **Currency** data type can be used instead of **Single** or **Double** if user needs to store values with up to four decimal places only.
- **Date/ Time:** It is used to store date and time. MS Access stores date in the standard mm/dd/yy format.
- **Currency:** It is used to store numbers as currency. The value in Currency field is rounded to 2 decimal places. The negative currency values are displayed in brackets.
- **AutoNumber:** It is used to generate the next number automatically when a new record is added. It creates a unique number for each record. The value starts from 1 and is incremented by 1 in each record.
- **Yes/No:** It is used to store Boolean value. The possible values in this field are **True** and **False**.

### 12.6.1 Field Property

**Field properties** are used to define how data will be entered, stored and displayed in MS Access. The properties of each field can be set in design view. The properties window is divided into two parts. The top pane is used for entering the field name, data type and an optional description of the field. The bottom pane is used for specifying the field properties.

Different field properties are as follows:

#### 1. Text Field Size

The **Field Size** is used to set the number of characters required in text or number field. The default field size for the text type is 50 characters. The field size can be limited to a certain number of characters if the value in the field is small. It saves disk space and prevents entry errors. The field size is set in exact characters for **Text** type.

#### 2. Format

**Format** is used to specify the format of data as it appears in the field. It has two parts for text and memo fields that are separated by a semicolon. The first part is used to apply to field and second applies to empty fields. Different types of formats in Access are as follows:

#### a. Text and Memo Format

Different formats for text and memo are as follows:

| Symbol | Explanation |
|---|---|
| @ | It indicates a required character or space |
| & | It indicates an optional character or space |
| < | It converts characters to lowercase |
| > | It converts characters to uppercase |
| \ | It adds characters to the end |
| @;"XYZ" | It displays the text inside double quotes if the user enters no value. |

Some examples of formats for text and memo are as follows:

| Format | Datasheet Entry | Display |
|---|---|---|
| @@@-@@@@ | 1234567 | 123-4567 |
| @@@-@@@& | 123456 | 123-456 |
| < | HELLO | hello |
| > | hello | HELLO |
| @\! | Hello | Hello! |
| @;"No Data Entered" | (blank) | No Data Entered |

## b. Number Format

Different formats for numbers are as follows:

| Format | Explanation |
|---|---|
| 0 | 0 is a placeholder that displays a digit or 0 if there is none. |
| # | # is a placeholder that displays a digit or nothing if there is none. |
| % | % multiplies the number by 100 and added a percent sign |

Different examples of formats for numbers are as follows:

| Format | Datasheet Entry | Display |
|---|---|---|
| ###,##0.00 | 123456.78 | 123,456.78 |
| $###,##0.00 | 0 | $0.00 |
| ###.00% | .123 | 12.3% |

## c. Currency Format

The currency formatting consists of four parts separated by semicolons. These parts are format for positive numbers, format for negative numbers, format for zero values and format for Null values.

| Format | Explanation |
|---|---|
| $##0.00;($##0.00)[Red];$0.00;"none" | Positive values will be normal currency format, negative numbers will be red in parentheses, zero is entered for zero values, and "none" will be written for Null values. |

## d. Date Format

The easiest way to apply a format is to select from **drop-down** list as follows:

| | |
|---|---|
| General Date | 06/19/94 5:34:23 PM |
| Long Date | June 19, 1994 |
| Medium Date | 19-Jun-94 |
| Short Date | 06/19/94 |
| Long Time | 5:34:23 PM |
| Medium Time | 5:34 PM |
| Short Time | 17:34 |

The user can also format the data according to his particular requirements.

Different date formats are as follows:

| Format | Explanation |
|---|---|
| d | It displays 1 or 2 characters for day. Its value can be from 1 to 31. |
| dd | It displays 2 characters for day. Its value can be from 01 to 31 such as 01. |
| m or mm | It displays month as a number. Its value can be from 1 to 12. |
| mmm | It displays month using three characters such as Jan, Feb etc. |
| mmmm | It displays full name of the month such as January, March etc. |
| / - | It displays separator character. |
| h | It displays hours. |
| n | It displays minutes. |
| s | It displayed seconds. |

Some examples of date formats are as follows:

| Format | Display |
|---|---|
| dddd","mmmm d","yyyy | Monday, January 1, 2001 |
| ddd","mmm ". " d", "'yy | Mon, Jan. 1, '01 |
| "Today is " dddd | Today is Monday |
| h:n:s: AM/PM | 12:00:00 AM |

### e. Yes/No Format

The Yes/No fields are displayed as check boxes by default on the datasheet. It can be changed to textbox by clicking **Lookup** tab and changing **Display Control** to a textbox. The formatting is designated in three sections separated by semicolons. The first section contains nothing but semicolon must be included. The second section specifies formatting for **Yes** values and third for **No** values as follows:

| Format | Explanation |
|---|---|
| ;"Yes"[green];"No"[red] | Prints "Yes" in green or "No" in red |

## 3. Default Value

In some cases, the value of all records in a certain field is same. A default value can be set in this case. The default value already appears in the field when the user enters data. So the user does not need to type the same value again and again. The property **Set the Default Value** is used to set default value for a field.

## 4. Indexes

The indexes are created to obtain and sort records faster in MS Access. The **Indexed** property is used to set an indexed field. The **Yes (Duplicates OK)** is selected if multiple entries of same data value are allowed. The **Yes (No Duplicates)** option prevents duplicates.

## 5. Field Validation Rules

The **Validation Rules** specify the criteria for the data entered in the field. A message can be displayed to the user if the data violates the rules set for the field. The **expression builder** ("...") button at the end of **Validation Rule** box is used to write validation rule. For example, a validation rule **<> 0** indicates that zero cannot be entered in the record. The rule **???** indicates that the data may consist of only three characters.

Following are some examples of commonly used operators:

| Expression | Meaning |
|---|---|
| >=Date () | A date that is either today's date or some date in the future. |
| BETWEEN 10 AND 100 | A value between 10 and 100. |
| "UK" OR"USA" | Match UK or USA |
| LIKE "K???" | Value must be four letters beginning with K. |
| "M" OR "F" | Entry must be M or F. |

## 6. Validation text

A property used to specify the message to be displayed to the user when a validation rule is violated

## 7. Input Masks

An input mask controls the value of a record and sets it in a specific format. It is similar to **Format** property but it displays the format on datasheet before data is entered. A phone number field can be formatted with input mask to accept ten digits as "(555) 123-4567". The blank field will look like (___) ___-____. It helps user in entering value in specific format. The following symbols can be used to create an input mask:

| Character | Applies To | Allows | Entry Required? |
|---|---|---|---|
| 0 | Character | Digits (0 to 9) only | Yes |
| 9 | Character | Digits (0 to 9)<br>Spaces | No |
| # | Character | Digits (0 to 9)<br>Spaces<br>+ or – signs | No |
| L | Character | Letters (A to Z) only | Yes |
| ? | Character | Letters (A to Z) only | No |
| A | Character | Letters (A to Z)<br>Digits (0 to 9) | Yes |
| A | Character | Letters (A to Z)<br>Digits (0 to 9) | No |
| & | Character | Any character or spaces | Yes |
| C | Character | Any character or spaces | No |
| \ | Character | Display next character as literal (for example, \C displays C). | |
| < | Character | Convert following characters to lowercase. | |
| > | Character | Convert following characters to uppercase. | |

Any characters that are not in the above list will be treated as literals - they will be displayed in the appropriate position in the field. Here are some sample input masks:

| Mask Type | Mask | Examples |
|---|---|---|
| Social Security Number | >000-00-0000 | 545-33-4882 |
| Zip Code | >LL 00000 | PK 10021<br>PU 35228 |
| Telephone Number | (+099) 099 00000099 | (+41 ) 020 78877999<br>(+1 ) 206 555246<br>(+092) 1 345123 |

### 8. Caption Property

All data types have a caption property. The **Caption** property is used to display a label to the field. It appears at the top of columns in table's datasheet view.

### 9. Required

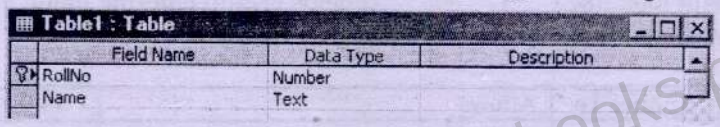
The **Required** property specifies whether the field is mandatory or not. If this property is set to **Yes**, the user has to enter data for the field to save the record. If this property is set to **No**, the user can leave the field blank. It does not apply to fields with **AutoNumber** data type as Access provides value for such data type.

## 12.7 Table Views in Access

Table view is a way of looking at the table. MS Access provides the following views:

### 1. Design View

The table view that is used to design the structure of a table is called **design view.** It is used to specify name, data types and description of fields. Primary key is also specified in this view. The structure of an existing table can also be changed in design view.




### 2. Datasheet View

The table view that is used to enter, delete or modify data in a table is called **datasheet view.** The table in this view is displayed in rows and columns. The name of each field is displayed at the top of the column.


### 3. PivotTable and PivotChart View

MS Access 2002 provides two additional views that provide a convenient way to display summary information from a table. The **PivotTable view** is similar to an excel pivot table and summarizes data about groups and records. **PivotChart view** creates a chart from the associated pivot Table view.

### Switching between Views

Following procedure is used to switch between table views:

- Click on **View** menu.
- Select the desired view from the menu. The table will be displayed according to the selected view option.

## 12.8 Methods of Modifying a Table

An existing table can be modified in the following ways:

- Adding Records
- Editing Records

- Deleting Records
- Inserting and Deleting Fields
- Resizing Rows and Columns
- Reordering Column
- Freezing and hiding columns.

## 1. Adding Record

The user can add new records to table in datasheet view. The new data is typed in the record that has an asterisk (*) on left side. It indicates the new record. The user can also click **new record** button at the bottom of datasheet to move to the last empty record to add the new record.

## 2. Editing Records

The user can edit records by placing the cursor in the record to be edited and making necessary changes. The arrow keys are used to move through the record grid. The **previous, next, first** and **last** record buttons at the bottom of datasheet are helpful in editing datasheet. There are various ways to edit records.

- To move to the next or the previous field, press **TAB** or **SHIFT+TAB**, respectively.
- To select or deselect the current field, press **F2**.
- To undo changes to the current field or record, press **ESC**.
- To replace the value in a field with the value of the same field in the previous record, press **CTRL+'**.

## 3. Deleting Records

The user can modify a table by deleting a record on a datasheet. The following procedure can be used to delete a record from the table:

- Placing cursor in any field of the record to be deleted.
- Select **Edit > Delete Record** from menu bar OR click **Delete Record** button on the datasheet toolbar.

## 4. Inserting, Deleting and changing Field name

A better method to add new fields is the use of design view. The design view provides more options to add fields quickly. The following procedure is used to add a new column in datasheet view:

- Highlight the column before which the new column is to be added by clicking its label at the top of the datasheet.
- Select **Insert > Column** from the menu bar.

The following procedure is used to delete a column in datasheet view:

- Select the column to be deleted by placing the cursor in it.
- Select **Edit > Delete Column** from the menu bar.

## Changing Field Name in Design View

- Open the table in design view.
- Click on the field name to be changed.
- Edit the text and type the new name for the field and press **Enter**.

## Changing Field Name in Datasheet View

- Open the table in datasheet view.

- Double click the field selector at the top of the column. The field name will be selected.
- Edit the text and type the new name for the field and press **Enter**.

OR

- Right click the field name. A popup menu will appear:
- Select **Rename column** option. The column will be selected.
- Edit the text and type the new name for the field and press **Enter**.

## 5. Resizing Rows and Columns

The height of rows on a datasheet can be changed. The user can drag the gray sizing line between row labels up and down with the mouse. The height of all rows is changed if there is any change in the height of any row.

The column width can also be changed. The user can drag the sizing line between columns. The user can double click on line expand the column according to the longest value in the column. Different columns on a datasheet can have different widths. The exact values can be assigned from **Format > Row Height** OR **Format > Column Width** from menu bar.

## 6. Reordering Columns

The order of columns in a table can be changed in Access. The change in columns order does not affect table and data stored in it. The procedure of reordering columns is as follows:

- Open the table in datasheet view.
- Select the column whose position is to be changed. A column is selected by clicking the field selector at the top of the column. Multiple columns can be selected by dragging the mouse over field selectors.
- Drag the selection to the left or right. As you move the pointer, a bold black line will indicate the place where the columns will be moved.
- Release the mouse button to place the selected columns at new place..

## 7. Freezing and hiding columns.

### Freezing Columns

The user can freeze the columns on an Access table. It helps the user to view certain columns easily if the datasheet has many columns and the required columns are not visible. The following procedure is used to freeze and column:

- Placing the cursor in any record in the column to be freezed.
- Select **Format > Freeze Columns** from the menu bar.

The option **Format > Unfreeze** can be used to unfreeze a single column.

### Hiding Columns

The columns can be hidden from datasheet temporarily. The hidden columns are not be deleted from the database. The following procedure is used to hide a column:

- Place the cursor in any record in the column to be hidden OR highlight multiple adjacent columns by clicking and dragging the mouse along the column headers.
- Select **Format > Hide Columns** from the menu bar.

The option **Format > Unhide Columns** can be used to unhide the columns. A window will appear displaying all fields in the table. The check boxes appear with all fields. The selected checkbox indicates that the field is visible. The user can check certain fields to unhide them and then click **Close** button.

## Practical 1

a. Create a new blank database with file name **Employees**.
b. Create a new table in **Design** view and add fields to the table as follows:

| Field Name | Data Type | Description |
|---|---|---|
| LastName | Text | Enter employee's surname |
| FirstName | Text | Enter employee's first name(s) |
| SocialSecurityNo | Text | Enter Social Security number |
| BirthDate | Date/Time | Enter birth date |
| JobTitle | Text | Title of Job |
| Department | Text | Department name |
| Salary | Currency | Annual salary |
| StartDate | Date/Time | Employment start date |
| FinishDate | Date/Time | Employment finished |
| SocialClub | Yes/No | In Social Club (yes/no)? |

c. Make the **SocialSecurityNo** field **Primary key** and save the table as **Employees**.
d. Open the table in **Datasheet view** and add five records in the table.

## Procedure

### a. Creating a Blank Database

1. Click **Start > Programs > Microsoft Access** to start MS Access.
2. Select **Blank Access Database** option.
3. Click **OK**. A dialog box will appear to input database name.
4. Type **Employees** in **File Name** box.
5. Select **My Documents** folder to save the database.
6. Click **Create**. A new database will be created and the **Database** window will appear.

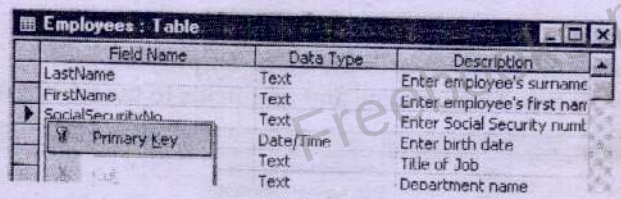
### b. Creating a Table

1. Click on the **Tables** object in the main database window.
2. Double click **Create table in Design view** in **Database** window. A new blank table will appear.
3. Type field name **LastName** in **Field Name** column.
4. Press **Tab** key to move to **Data Type** column and select the data type **Text**.
5. Press **Tab** key to move to **Description** column and type **Enter employee's surname** as a comment.
6. Press Tab key to move to **Field Name** for the next field.
7. Repeat steps 2 to 5 until all fields have been defined.

| Field Name | Data Type | Description |
|---|---|---|
| LastName | Text | Enter employee's surname |
| FirstName | Text | Enter employee's first name(s) |
| SocialSecurityNo | Text | Enter Social Security number |
| BirthDate | Date/Time | Enter birth date |
| JobTitle | Text | Title of Job |
| Department | Text | Department name |
| Salary | Currency | Annual salary |
| StartDate | Date/Time | Employment start date |
| FinishDate | Date/Time | Employment finished |
| SocialClub | Yes/No | In Social Club (yes/no)? |

### c. Assigning a Primary Key

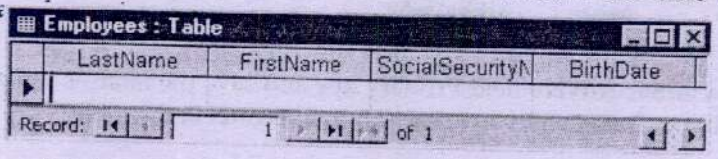
1. Place cursor in **SocialSecurityNo**.
2. Click **Edit > Primary key OR** click **primary key** icon on **standard** toolbar. OR
3. Right click SocialSecurityNo and choose **Primary Key** from the pop up menu.



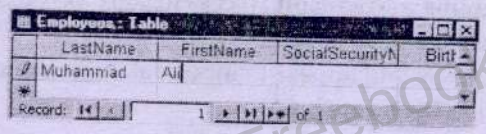

4. Click **File > Save**. The **Save As** dialogue box will appear.
5. Enter **Employees** in **Table Name** and click **OK**. The table will be saved.

## d. Adding Records

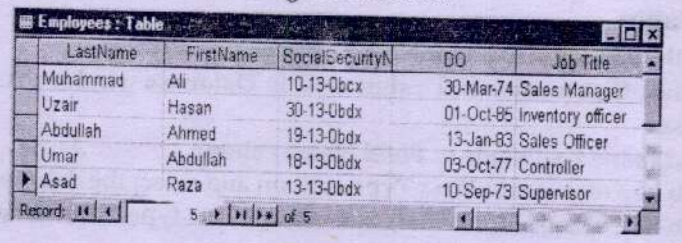
1. Double click **Employees** table in **Database** window. The table will appear in da




2. Type "Muhammad" in **LastName** field.
3. Press tab to move to next column and type "Ali".




4. Enter data in all the columns in same way.
5. Repeat step 2 to 4 to add remaining four records.




## Practical 2

a. Open **Employees** table used in Practical 1 and change **Field Size** property of **SocialSecurityNo** to 11.
b. Change **Decimal Places** property of **Salary** field to 0 (zero).
c. Choose **Short Date** from **Format** property box of **BirthDate** field.
d. Add a **caption** to each field as shown below and save changes:

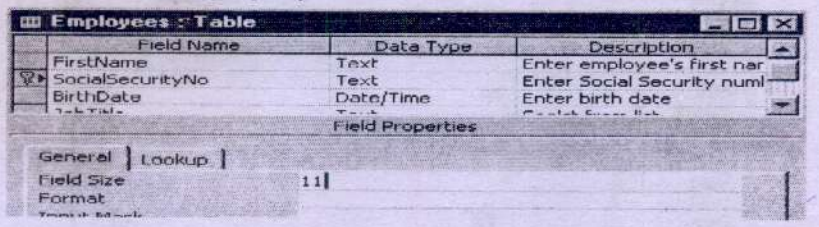
| Field | Caption | Field | Caption |
|---|---|---|---|
| LastName | Surname | Department | Department |
| FirstName | Forename(s) | Salary | Salary |
| SocialSecurityNo | Social Security No | StartDate | Start Date |
| BirthDate | DOB | FinishDate | Finish Date |
| JobTitle | Job Title | SocialClub | Social Club Member? |

e. Set **Department** field to **Required**.
f. Change date format of **DOB** to **Medium date**.

## Procedure

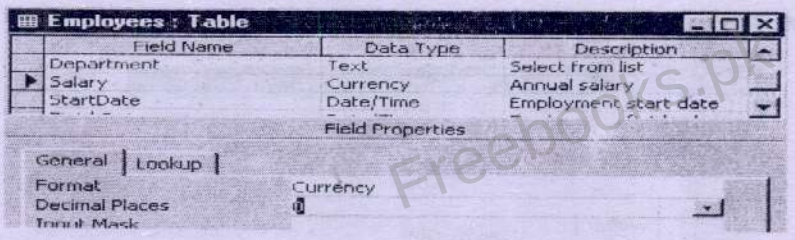
### a. Changing the Field Size Property

1. Open a table in **design view**.
2. Select **SocialSecurityNo** field.
3. Type 11 in **Field Size** property box.



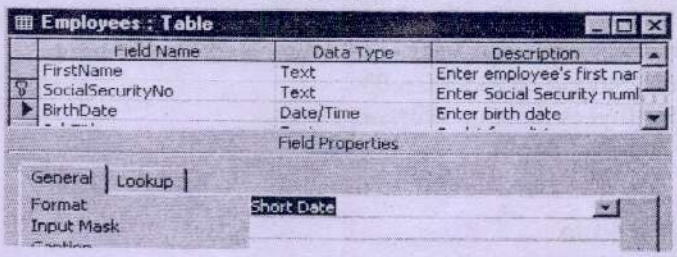

### b. Change the Decimal Places Property

1. Select **Salary** field from table design view.
2. Select 0 in **Decimal places** property box.



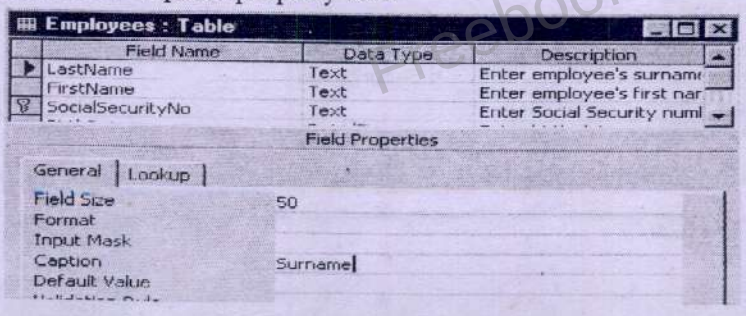

### c. Changing Date Format

1. Select **BirthDate** field from table **design** view.
2. Choose **Short date** from **Format** property box.




### d. Adding Caption

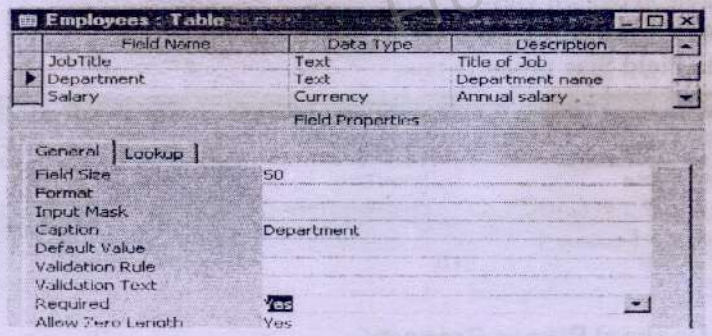
1. Select **LastName** field from table design view.
2. Type **Surname** in **Caption** property box.




3. Repeat step 1 to 2 to add captions to all other fields.

### e. Setting Required Property

1. Select **Department** field and choose **Yes** from **Required** property box.




### f. Changing Date Format to Medium

1. Select **BirthDate** field.
2. Choose **Medium Date** from **Format** property box.

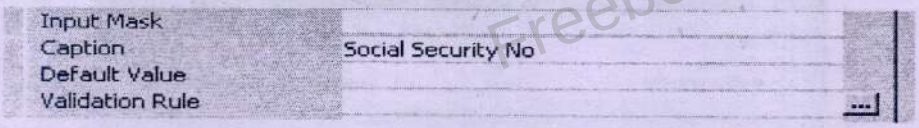

## Practical 3

a. Open **Employee** table used in Practical 1 and apply **validation rule** on **Social security** that it must not be 0. It should display 'Not valid No' in case of wrong data.
b. Hide **Lastname**, **Firstname**, **SocialSecurityno** and **Department**.
c. Delete record having **DOB** 13-Jan-83.
d. Rename **Employees** table to 'Workers Data'.
e. Freeze the **Salary** column.

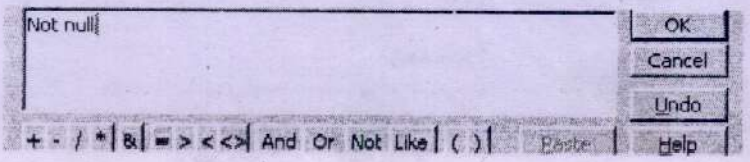
## Procedure

### a. Apply Validation Rule

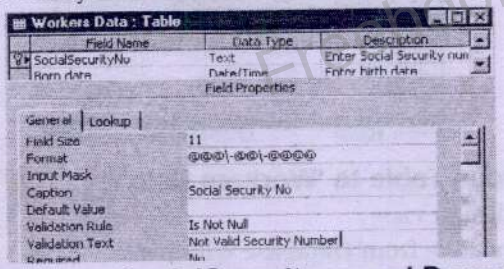
1. Open a table in **design** view.
2. Select **SocialSecurityNo** field.
3. Click **Expression Builder Button** (three dots) on the right of **Validation Rule** property box.




4. **Expression Builder** dialog box will appear. Make expression **Not null** in the text box.

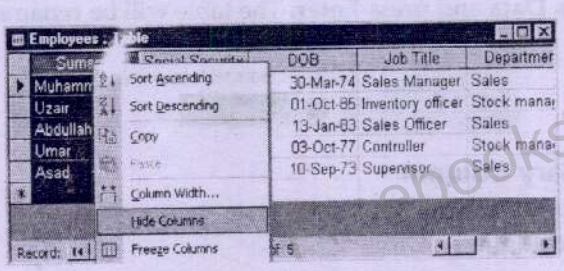


5. Click **OK** to apply the rule.
6. Type "Not Valid Security Number" in **Validation Text** property box.




## b. Hide Lastname, Firstname, SocialSecurityno and Department.

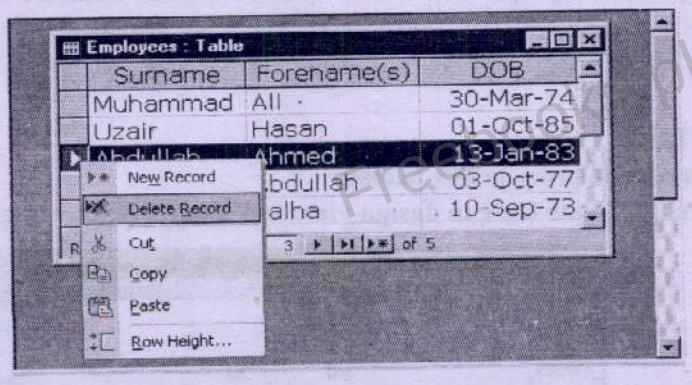
1. Open a table in **datasheet** view
2. Select **Lastname** column from table.
3. Right click and choose '**Hide columns**' option from popup menu. The column will disappear.




4. Use the same procedure to hide remaining columns.

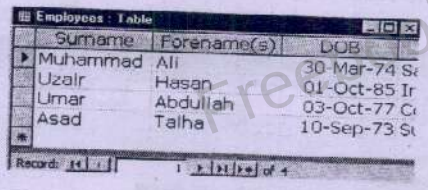

## c. Deleting Record having DOB 13-Jan-83.

1. Select the record having DOB 13-Jan-83.



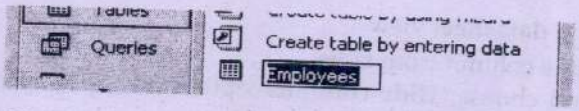

2. Right click it and select **Delete Record** option. A confirmation box will appear.

3. Click **Yes**. The record will be deleted from table.

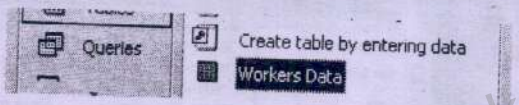



### d. Renaming Employees Table to 'Workers Data'.

1. Open a table in design view.
2. Select **Employees** table from **Database** window.
3. Click **Edit > Rename**. The cursor will blink in **Employees** table name.

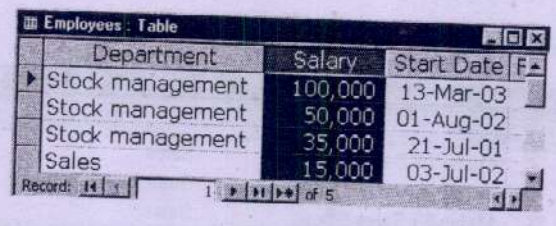



4. Type **Workers Data** and press **Enter**. The table will be renamed.




### e. Freezing the Salary Column

1. Open a table in **datasheet** view.
2. Select **Salary** column.




3. Click **Format > Freeze Columns**. The column will be freezed.

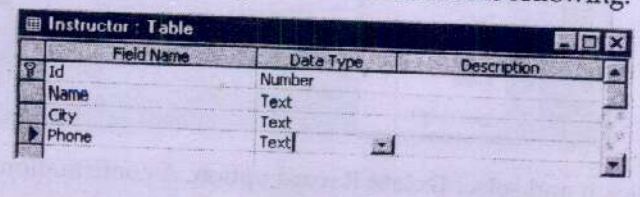
## Practical 4

a. Create a table named 'Instructor' (Id, Name, City, Phone).
b. Add Input mask property to **Phone** field so it takes numbers in (+092) 41 733474 format.
c. Add five records in the table.

## Procedure

### a. Creating a Table

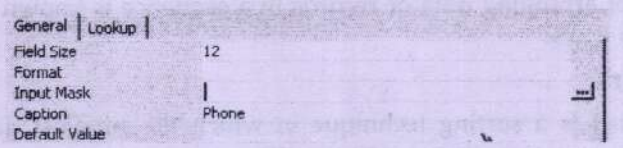
1. Create a new database in **MS Access**.
2. Double click **Create table in design view** and create the following:



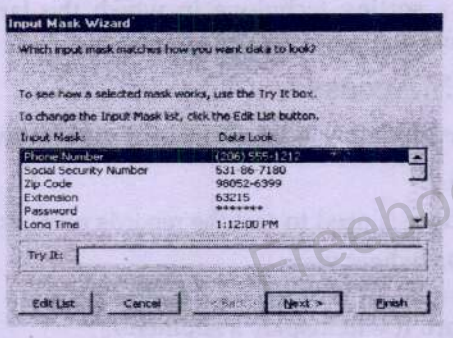
3. Right click **Id** field and select **Primary Key** from popup menu.
4. Click **Ctrl+S**. A dialog box will appear.
5. Type **Instructor** as table name and press **OK**. The table will be saved.

## b. Apply Input Mask Property to Phone Field

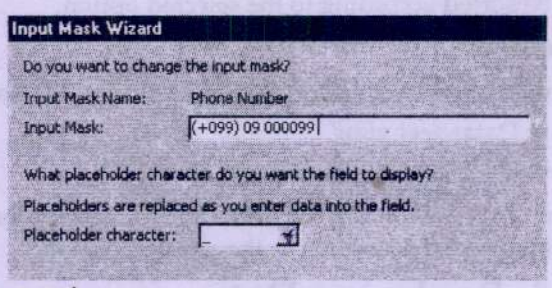
1. Open the table in **Design view**.
2. Select the **Phone** field.
3. Click in white space following **Input Mask** under **General** properties.



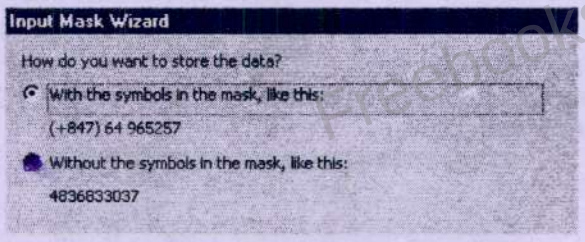

4. Click **Build** button ...|. The **Input Mask Wizard** dialogue box will appear.
5. Select **Phone Number** input mask from the list and click **Next** to proceed.




6. Type **(+099) 09 000099** in **Input Mask** text box and select – from **Placeholder** drop down list.




7. Click **Next** to proceed.
8. Select **With the symbols in the mask, like this:** radio button and press **Next >**.




9. Click **Finish**.

## c. Adding Records

1. Open a table in **Datasheet View** of table and add the following records:

Instructor : Table

| Id | Name | City | Phone |
|---|---|---|---|
| 1 | Ali | FSD | (+092) 41 733474 |
| 2 | Asad | LHR | (+092) 42 070900 |
| 3 | Umar | LHR | (+092) 42 076766 |
| 4 | Uzair | FSD | (+092) 41 670980 |
| 5 | Hamza | LHR | (+092) 41 000963 |
| 0 | | | |

Record: 5 of 5

## 12.9 Sorting

The process of arranging data or records in a sequence is known as **sorting**. The data can be sorted in two ways:

### 1. Ascending Sort

**Ascending sort** is a sorting technique in which the smallest data is placed at first position and the largest data is placed at last position.

### 2. Descending Sort

**Descending sort** is a sorting technique in which the largest data is placed at first position and the smallest data is placed at last position.

### 12.9.1 Sorting in MS Access

The records in a table can be sorted in **Datasheet view** in two ways:

### 1. Sorting using Menu

The following procedure is used to sort the records using menu:

1. Open the table in **Datasheet view**.
2. Select the field according to which data will be sorted.
3. Select **Records > Sort.** It will display a submenu.
4. Choose **Sort Ascending** or **Sort Descending**.

The records will be sorted according to the selected option.

### 2. Sorting using Toolbar

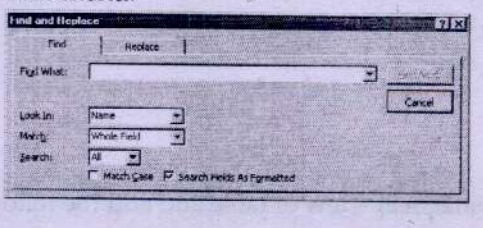
The following procedure is used to sort the records using toolbar:

1. Open the table in **Datasheet view**.
2. Select the field according to which data will be sorted.
3. Select **Sort Ascending** or **Sort Descending** icon on the toolbar.
4. The records will be sorted according to the selected option.

## 12.10 Finding & Replacing Data

Access provides facility to find the required data easily. The following procedure is used for finding data:

1. Open the table in datasheet view.
2. Place the cursor in a field to find data.
3. Choose **Find** from **Edit** menu.



Find and Replace

Find | Replace

Find What:

Find Next | Cancel

Look In: Name

Match: Whole Field

Search: All

Match Case | Search Fields As Formatted

4. Enter the data to find in **Find What** textbox.
5. Select any option from **Match** list box. It is used to match the entered with whole field, any part of field or start of field.
6. Click **Find Next** button. If the data entered in **Find What** textbox is found in the field, the cursor will move to that record. If the data is not found, a message will appear showing that the data is not found.

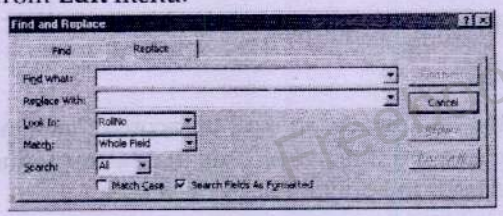
## Replacing data

MS Access provides facility to find and replace data easily. Replacing data consists of two steps:

1. Finding the data that matches the search string
2. Replacing the matched data with the given replace string

The following procedure is used for replacing data:

3. Open the table in datasheet view.
4. Place the cursor in a field to find data.
5. Choose **Replace** from **Edit** menu.




6. Enter the data to find in **Find What** textbox.
7. Select any option from **Match** list box. It is used to match the entered with whole field, any part of field or start of field.
8. Enter the data to replace in **Replace With** textbox.
9. Click **Find Next** button. If the data entered in **Find What** textbox is found in the field, the cursor will move to that record.
10. Click on the **Replace** button to replace the search string. OR click on **Replace All** button to replace all matching data in the table.

## 12.10.1 Options in Find and Replace Dialog Box

The **Find and Replace** dialog box is used to find and replace data in tables. It searches data using different criteria given by user. Different options of the dialog box are as follows:

- **Find What:** This textbox is used to enter the data that is to be searched. The user can enter any text including alphabetic, numbers and special characters.
- **Look In:** This list box is used to specify the area in which search will be performed. It contains the selected field name and table name. If the user selects the field name, search is done in the selected field only. If the user selects table name, search is done in the whole field.
- **Match:** The **Match** list box is used to specify the way the given data will be matched with the values in the field. It provides three options:
    - **Whole Field:** It is used to find only those fields that exactly match the search string.
    - **Any Part of Field:** It is used to find fields that contain search string in any part of the field. If user enters 12 in **Find What** box, it will match any string like 1234, 121, 512.

- **Start of Field:** This option is used to find the fields that begin with the search string. The search string 12 will match 1234, 124, 12 but not 312.

- **Search:** It is used to specify the direction of search. It provides following options:
  - **Up:** If this option is used, the search starts from the current record and moves to the beginning of the table.
  - **Down:** If this option is used, the search starts from the current record and moves to the end of the table.
  - **All:** If this option is used, the search is performed on all records from the beginning to the end of table.
- **Match Case:** This option is used to specify case sensitivity of search. If it is selected, search is case sensitive. If the checkbox is not selected, the search is case insensitive.

## 12.11 Filter

A **filter** is used to extract the records that match a set of criteria. Filters are used with opened tables.

### 12.11.1 Types of Filters

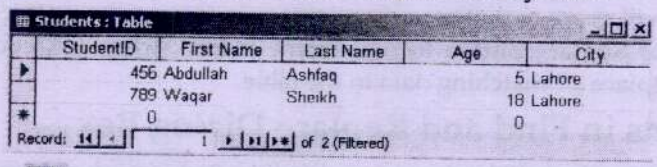
Different types of filters are as follows:

#### 1. Filter by Selection

The **filter by selection** feature is used to filter records that contain identical data values in a given field. For example, it can filter all records that have a value "Usman" in **name** field. The following procedure is used to apply Filter by Selection:

1. Place the cursor in the field according to which the filter is to be applied.
2. Click **Filter by Selection** button on toolbar OR select **Records > Filter > Filter By Selection** from the menu bar.

The following example displays the records of Student table. The filter is applied on **City** field. It displays the records with the value "Lahore" in **City** field.



Students : Table

| StudentID | First Name | Last Name | Age | City |
|---|---|---|---|---|
| 456 | Abdullah | Ashfaq | 5 | Lahore |
| 789 | Waqar | Sheikh | 18 | Lahore |
| 0 | | | 0 | |

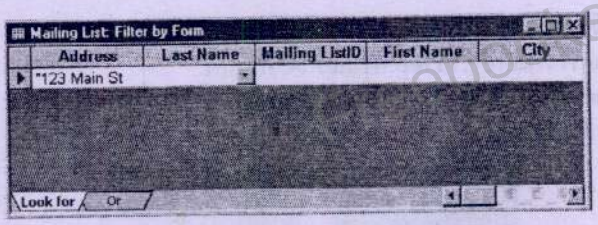
Record: 1 of 2 (Filtered)

#### 2. Filter by Form

The **Filter by Form** is useful if the table is large and it is difficult for the user to find the record that contains the value according to which the filter is to be applied. This method creates a blank version of the table with drop-down menus. Each field contains the values found in the records of that field.

The following procedure is used to apply **Filter by Form**:

1. Click the field to enter the filter criteria under default **Look for** tab of **Filter by Form** window.
2. Click **Or** tab at the bottom of window to specify an alternate criteria if records may contain one of two specified values.
3. Select another criteria from drop-down menu. More **Or** tabs will appear after one criteria is set to add more alternate criteria for the filter.
4. Click **Apply Filter** button on toolbar after selecting all criteria to filter.




The following methods can be used where the drop-down menu appears instead of selecting an absolute value.

| Filter by Form | |
|---|---|
| **Format** | **Explanation** |
| **Like "*Street"** | Selects all records that end with "Street" |
| **<="G"** | Selects all records that begin with the letters A through G |
| **>1/1/00** | Selects all dates since 1/1/00 |
| **<> 0** | Selects all records not equal to zero |

## Practical 5

Perform the following task using the previous employee table

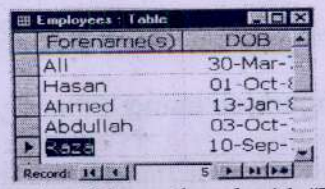
a. Find record having name Raza & Replace it with Talha.
b. Sort **Department** column in descending order.
c. Filter record by 'Stock management' department using **'Filter By Selection'**
d. **Remove** the above applied filter
e. Filter record by 'Sales manger' job using **'Filter By Form'**
f. **Remove** 'Filter By Selection'

## Procedure

### a. Finding Record having Name Raza & Replace it with Talha

1. Open a table in **datasheet** view.
2. Select **Edit > Replace**. The **Find and Replace** dialog box will open.
3. Type "Raza" in **Find What** box.
4. Select **Employees: Table** from **Look in** list and **Whole Field** from **Match** list.
5. Type "Talha" in **Replace With** text box.

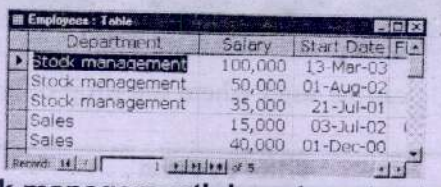

6. Click **Find Next** button. The text Raza will be highlighted in **Employees** table.




7. Click **Replace** button. "Raza" will be replaced with "Talha" in Employees table.

### b. Sorting Department Column in Descending Order

1. Place pointer anywhere in **Department** column.
2. Click **Records > Sort > Descending Sort**. The data will be sorted according to **Department** column.



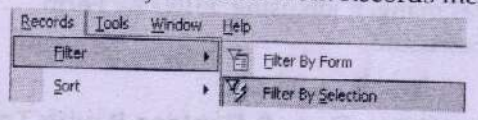
| Department | Salary | Start Date |
|---|---|---|
| Stock management | 100,000 | 13-Mar-03 |
| Stock management | 50,000 | 01-Aug-02 |
| Stock management | 35,000 | 21-Jul-01 |
| Sales | 15,000 | 03-Jul-02 |
| Sales | 40,000 | 01-Dec-00 |

### c. Filter Record by 'Stock management' department using 'Filter By Selection'

1. Place the cursor in Stock management field.

| DOB | Job Title | Department | Salary |
|---|---|---|---|
| 30-Mar-74 | Sales Manager | Sales | 40,000 |
| 01-Oct-85 | Inventory officer | Stock management | 100,000 |
| 13-Jan-83 | Sales Officer | Sales | 15,000 |
| 03-Oct-77 | Controller | Stock management | 50,000 |
| 01-Oct-85 | Sales Manager | Stock management | 35,000 |
| | | | 0 |

2. Choose Filter and then Filter By Selection from Records menu.

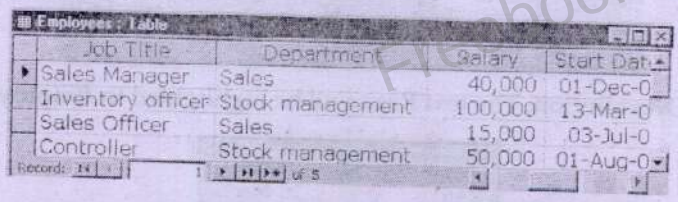



3. Data will be filtered according to Stock management.

| DOB | Job Title | Department | Salary |
|---|---|---|---|
| 01-Oct-85 | Sales Manager | Stock management | 35,00 |
| 03-Oct-77 | Controller | Stock management | 50,00 |
| 01-Oct-85 | Inventory officer | Stock management | 100,00 |

### d. Remove the above applied Filter

1. Select **Remove Filter/Sort** from **Records** menu.
2. Filter will be removed.



| Job Title | Department | Salary | Start Dat |
|---|---|---|---|
| Sales Manager | Sales | 40,000 | 01-Dec-0 |
| Inventory officer | Stock management | 100,000 | 13-Mar-0 |
| Sales Officer | Sales | 15,000 | 03-Jul-0 |
| Controller | Stock management | 50,000 | 01-Aug-0 |

### e. Filter record by 'Sales manger' job using 'Filter By Form'

1. Open table in **Datasheet** view.
2. Choose Filter and then **Filter By Form** from **Records** menu.

3. Click in 'Job Title' field, a drop down arrow will appear.
4. Press drop down arrow and Select 'Sales Manager' from drop down list.


5. Click **Apply Filter** icon from Standard toolbar.


6. Records will be filtered according to Sales Manger.


## f. Remove 'Filter By Selection'

1. Press Remove Filter icon from Standard toolbar to remove the filter.


2. Filter will be removed.

CHAPTER 13

# Relationships in MS Access

## Chapter Overview

# 13.1 Relationship

A **relationship** is a logical connection between different tables. A relationship is established on the basis of interaction among these tables. The relationship is established by connecting one or more fields of two tables. The fields used to connect two tables normally have same name, data type and size.

## 13.1.1 Types of Relationships

Different types of relationships are as follows:

### 1. One-to-One Relationship

One-to-one relationship exists between two tables when:

- For each record in first table, there is only one record in the second table.
- For each record in second table, there is only one record in the first table.

An example of one-to-one relationship is as follows:

Primary Key — One-to-one relationship — Primary Key

| EmpID | Name | Joining | Salary |
|---|---|---|---|
| 01 | Ali | 01/04/2002 | 13000 |
| 02 | Ahmad | 02/04/2004 | 11000 |
| 03 | Nadeem | 10/05/2006 | 8000 |

Employee Table

| EmpID | Loan | Years |
|---|---|---|
| 01 | 500000 | 5 |
| 02 | 100000 | 5 |
| 03 | 50000 | 2 |

Loan Table

The above example has two tables **Employee** and **Loan**. The **Employee** table contains the data of employees and **Loan** table contains data about loans. Both tables contains **EmpID** field as primary key. Each employee can borrow loan once so the tables are joined with one-to-one relationship. It means that for each record in Employee table, there is only one record in Loan table and vice versa. Each EmpID occurs once in both tables.

### 2. One-to-Many Relationship

One-to-many relationship exists between two tables when:

- For each record in first table, there are one or more records in the second table.
- For each record in second table, there is only one record in the first table.

An example of one-to-many relationship is as follows:

Primary Key — One-to-many relationship — Composite Primary Key

| RollNo | Name | Address |
|---|---|---|
| 1 | Ali | Faisalabad |
| 2 | Ahmad | Lahore |
| 3 | Nadeem | Peshawar |

Student Table

| RollNo | Subject | Marks |
|---|---|---|
| 1 | English | 52 |
| 1 | Math | 77 |
| 1 | Computer | 64 |
| 2 | English | 58 |
| 2 | Math | 69 |
| 2 | Computer | 49 |
| 3 | English | 82 |
| 3 | Math | 98 |
| 3 | Computer | 86 |

Marks Table

The above example has two tables **Student** and **Marks**. The **Student** table contains data of the students and its primary key field is **RollNo**. The **Subject** table contains the marks of the students in different subjects and its primary key consists of two fields **RollNo** and **Subject**. The **RollNo** field also works as foreign key in **Marks** table. Both tables are joined with one-to-many relationship. It means that for each record in **Student** table, there can be one or more records in **Subject** table. There are more records in **Subject** table against one record in **Student** table because one student studies many subjects.

## 3. Many-to-Many Relationship

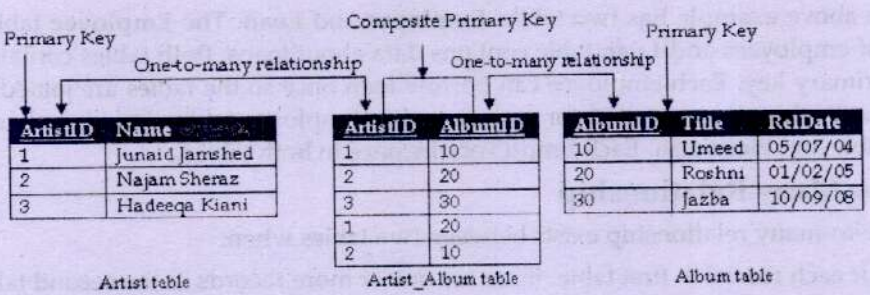
Many-to-many relationship exists between two tables when:

- For each record in first table, there are one or more records in the second table.
- For each record in second table, there are one or more records in the first table.

Many-to-many relationship is more complex than other types of relationships. It is a collection of two one-to-many relationships. The following steps are performed to create a many-to-many relationship:

1. Create the two tables to be connected using many-to-many relationships.
2. Create a third table known as **junction table**.
3. Insert the primary fields of two tables in the junction table and create a composite primary key that consists of both fields in junction table.
4. Create one-to-many relationship of both tables with junction table.

Many-to-many relationship will be established between both tables after completing the above steps. An example of many-to-many relationship is as follows:




The above example has three tables. The **Artist** table contains data of different artists and its primary key is **ArtistID**. The Album table contains data of different albums. One artist may work in many albums and one album may contain songs of different artists. Many-to-many relationship should be established between these tables. A junction table **Artist_Album** is created for this purpose. It consists of the primary fields of **Artist** table and **Alubum** table. The junction table is connected with both tables using one-to-many relationship.

**Practical 1**

a. Create a new blank database with file name student.
b. Create two tables; one with the name
   StdMaster (RegNo, Name, Address)
   And other with the name
   StdDetail (RegNo, Subject, Marks)
c. Insert at least three records in each table.
d. Create relationship between these tables.

### a. Creating a Blank Database

1. Click **Start > Programs > Microsoft Access** to start MS Access.
2. Select **Blank Access Database** option.
3. Click **OK**. A dialog box will appear to input database name.
4. Type **Student** in File Name box.
5. Select My Documents folder to save the database.
6. Click **Create**. A new database will be created and the **Database** window will appear.

### b. Creating a Table

### Creating StdMaster Table

1. Create table in design view.
2. Select RegNo as Primary key.
3. Save the table as StdMaster.

**StdMaster : Table**

| Field Name | Data Type |
|---|---|
| RegNo | Number |
| Name | Text |
| Address | Text |

### Creating StdDetail Table

1. Create table in design view.
2. Select RegNo and Subject as Primary Key.
3. Save the table as StdDetail.

**StdDetail : Table**

| Field Name | Data Type |
|---|---|
| RegNo | Number |
| Subject | Text |
| Marks | Number |

### a. Inserting at least three records in each table.

### Adding Records to StdMaster

1. Switch to Datasheet view of table and enter the data.

| RegNo | Name | Address |
|---|---|---|
| 96-AG-1940 | Abdullah | Faisalabad |
| 96-AG-1989 | Nadeem | Lahore |
| 96-AG-1991 | FAIQ | Peshawar |

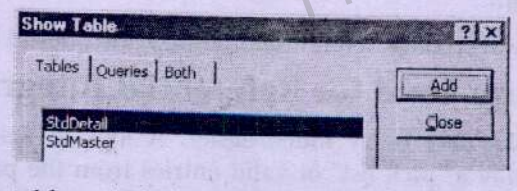
### Adding Records to StdDetail

1. Switch to Datasheet view of table and enter the data.

| RegNo | Subject | Marks |
|---|---|---|
| 96-AG-1940 | English | 52 |
| 96-AG-1940 | Math | 77 |
| 96-AG-1940 | Computer | 64 |
| 96-AG-1989 | English | 58 |
| 96-AG-1989 | Math | 69 |
| 96-AG-1989 | Computer | 49 |
| 96-AG-1991 | English | 82 |
| 96-AG-1991 | Math | 98 |
| 96-AG-1991 | Computer | 86 |

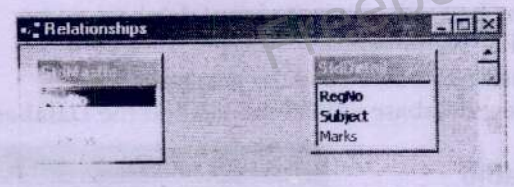
### Creating Relationship between Tables

- Select **Relationships** from **Tools** menu. The following dialog box will appear:

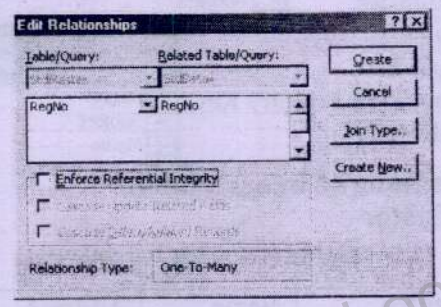



- Select **StdMaster** table and click **Add** button.

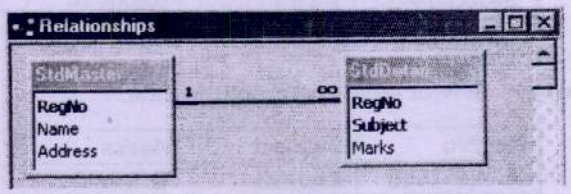
- Select **StdDetail** table and click **Add** button.
- Click **Close** button to close dialog box. **Relationships** window will appear as follows:




- Click on **RegNo** in **StdMaster** table, drag it on **RegNo** in **StdDetail** table and release the mouse. A dialog box will appear as follows:




- Select "Enforce Referential Integrity" checkbox and click **Create** button. Both tables will be joined by one-to-many relationship as follows:




## 13.1.2 Cascade Update Related Fields

The **Cascade Update Related Field** means that if the value of primary key field of any record is changed in parent table, MS Access automatically updates the corresponding values to the new value in all related records. For example, if the record of RollNo 3 is updated in the Student table, all related records of RollNo 3 are updated automatically in Result table.

## 13.1.3 Cascade Delete Related Records

The **Cascade Delete Related Field** means that if a record is deleted from the Parent table, MS Access automatically deletes all corresponding records in the child tables. For example, if the record of RollNo 3 is deleted from Student table, all related records of RollNo 3 are deleted automatically.

**Note:** Both options overcome Referential Integrity and should be used with care.

## 13.1.4 Use a Lookup Field for Referential Integrity

Lookup fields make data entry much easier. A lookup field is typically used in a foreign key field to display a "pick-list" of valid entries from the parent table. You can also use lookup fields in non-key fields to assist user entry. For example, you may create a lookup field to help the user select the name of a country or job title.

A lookup list is displayed in a drop-down list box (or combo box) on forms and datasheets. Values for the lookup field can be stored in another table or query or stored as part of the field definition by typing entries into the list.

## Practical 2

a. Create the following tables in Access:

**DEPT** (DeptNo, Dname, Loc)

**EMP** (EmpId, EmpName, Job, Sal, DeptNo)

b. Insert four records in Dept table.

c. Add Lookup to Deptno field of EMP table

## Procedure

### a. Create Dept table (DeptNo, Dname, Loc)

1. Create a table in design view.
2. Select *DeptNo* as **Primary Key.**
3. Save the table as *Dept*.

Dept : Table

| Field Name | Data Type | Description |
|---|---|---|
| DeptNo | Number | |
| Dname | Text | |
| Loc | Text | |

### Create 'Emp' table (EmpId, EmpName, Job, Sal, DeptNo)

1. Create a table in design view.
2. Select *EmpId* as **Primary Key**.
3. Save the table as *Emp*.

Emp : Table

| Field Name | Data Type | Description |
|---|---|---|
| EmpId | Number | |
| EmpName | Text | |
| Job | Text | |
| Sal | Text | |
| DeptNo | Number | |

### b. Add Records to Dept Table

1. Switch to Datasheet View of table & enter data for four records:

Dept : Table

| DeptNo | Dname | Loc |
|---|---|---|
| 10 | Accounting | FSD |
| 20 | Research | KHI |
| 30 | Sales | FSD |
| 40 | Operations | LHR |
| 0 | | |

Record: 4 of 4

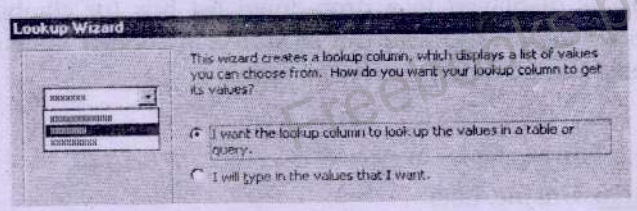
### c. Add Lookup field to Dept no in EMP Table

1. Open **Emp** table in **Design View**.
2. Select **Deptno** field and choose **Lookup Wizard** from **Data Type** column. **Lookup Wizard** is started.

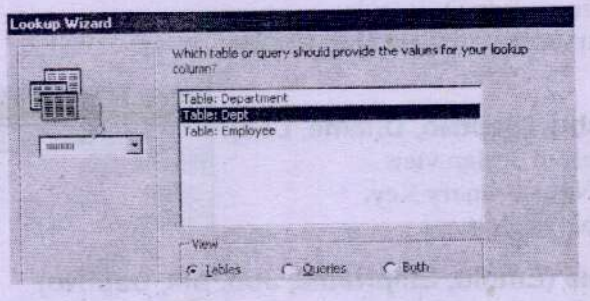
| Field Name | Data Type |
|---|---|
| EmpName | Text |
| Job | Text |
| Sal | Text |
| DeptNo | Number |

Text
Memo
Number
Date/Time
Currency
AutoNumber
Yes/No
OLE Object
Hyperlink
Lookup Wizard...

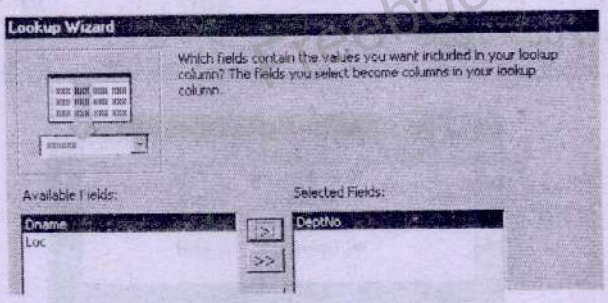
3. Choose the radio button saying I want the lookup column to look up the values in a table or query and press **Next >**.



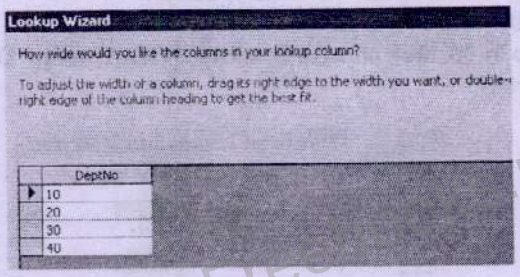

4. Select **Dept** table as data source for the lookup & click **Next >**.



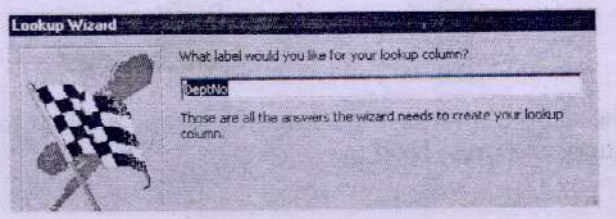

5. From the **Available Fields:** list, select *D*eptno and click the select button **>**.
6. Press **Next >** button to proceed.



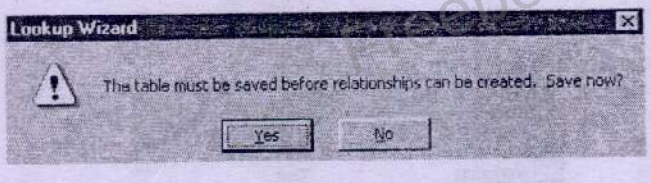

7. Click-and-drag the right edge of column to adjust column size to best fit and press **Next >**.



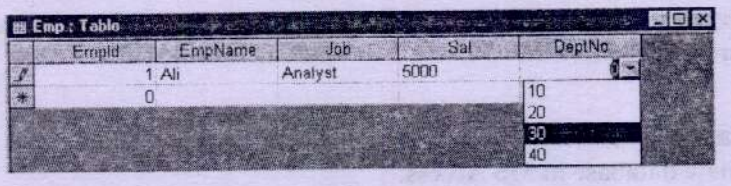

8. Lookup Wizard displays final dialogue box, prompting for a **label for the field**.



9. Leave the title as it is and click **Finish.**
10. Access prompts you to save changes to the table in order to create the relationship.




11. Click **Yes**.
12. The **Lookup** is created and a relationship is defined between the two tables.
13. Switch to **Datasheet View** to see the **Lookup column.**




## 13.2 Query

Query is a statement that extracts specific information from database. A query is created by specifying fields to display from a table or another query. It can also specify criteria on one or more fields for extracting data.

### Uses of Query

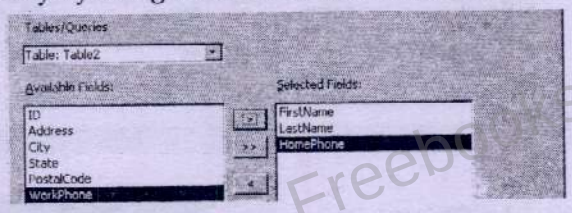
A query can be used for the following purposes:

- Extract records according to the specified criteria.
- Choose the fields to display in the result.
- Sort the records in a specific order.
- Calculate fields and summarize data.

## 13.3 Creating Query with Simple Query Wizard

The **Query Wizard** in MS Access helps the user to create a select query easily. It consists of simple steps to create query. The following procedure is used to create a select query using query wizard:

1. Click **Create query by using wizard** icon in database window to start query wizard.




2. Select the fields to be included in the query from the first window by selecting the table from **Tables/Queries** menu.
3. Select the fields by clicking > button to move the field from **Available Fields** list to **Selected Fields** OR click double arrow button >> to move all of the fields to **Selected Fields**.
4. Select another table or query to choose from more fields and repeat the process of moving them to **Selected Fields** box.

5. Click **Next >** when all of the fields have been selected.
6. Enter the name for the query and click **Finish**.

## Practical 3

a. Create a table called **Workshop Registration**.

Workshop Registration : Table

| Field Name | Data Type |
|---|---|
| RegistrationID | AutoNumber |
| Workshop | Text |
| FirstName | Text |
| LastName | Text |
| DepartmentName | Text |

b. Insert ten records in the table.

c. Create a query that shows First Name, Last Name and Workshop for workshop participants.

## Procedure

### a. Creating a Table

1. Create a new database in **MS Access**.
2. Double click **Create table in design view** and create the following:

Workshop Registration : Table

| Field Name | Data Type |
|---|---|
| RegistrationID | AutoNumber |
| Workshop | Text |
| FirstName | Text |
| LastName | Text |
| DepartmentName | Text |

3. Right click **RegistrationID** field and select **Primary Key** from popup menu.
4. Click **Ctrl+S**. A dialog box will appear.
5. Type **Workshop Registration** as table name and press **OK**. The table will be saved.

### b. Adding Records

1. Switch to **Datasheet View** of table and add the following records:

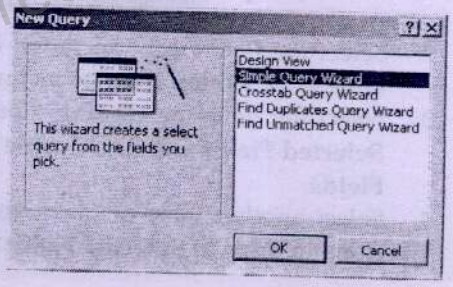
Workshop Registration : Table

| RegistrationID | Workshop | First Name | Last Name | Dept Name |
|---|---|---|---|---|
| 1 | Access | Adnan | Khalid | Economics |
| 2 | Access | Usman | Khalid | Psychology |
| 3 | Access | Javed | Ali | Psychology |
| 4 | Excel | Usman | Mahmood | Education |
| 5 | Excel | Abdullah | Ali | Education |
| 6 | Excel | Yasin | Anjum | Education |
| 7 | Powerpoint | Anjum | Manzoor | ISAT |
| 8 | Powerpoint | Sarah | Ali | ISAT |
| 9 | Word | Amjad | Ali | FM |
| 10 | Word | Jamil | Hamid | FM |
| (AutoNumber) | | | | |

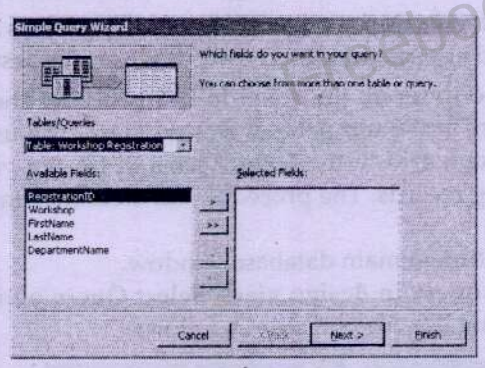
Record: 10 of 10

### c. Creating a Query

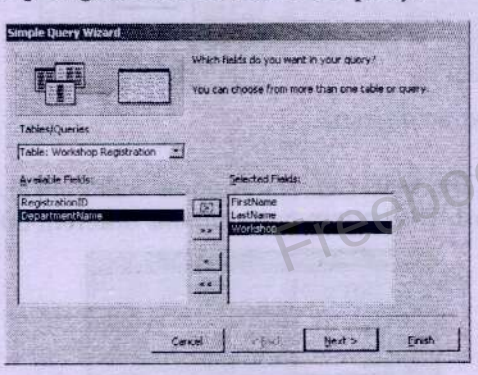
1. Click on **Queries** under **Object** in the **Database** window
2. Click **New** button on **Database** window toolbar. **New Query** dialog box will appear.
3. Select **Simple Query Wizard** and click **OK**. The **Simple Query Wizard** will appear.
4. Select **Tables/Queries** list. A list of available tables and queries will appear.



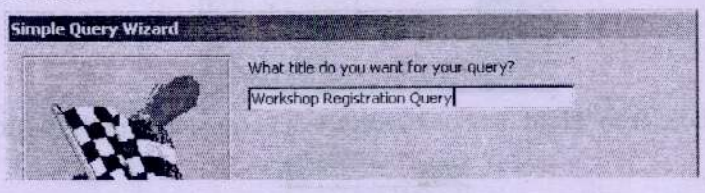
5. Select **Table: Workshop Registration.**



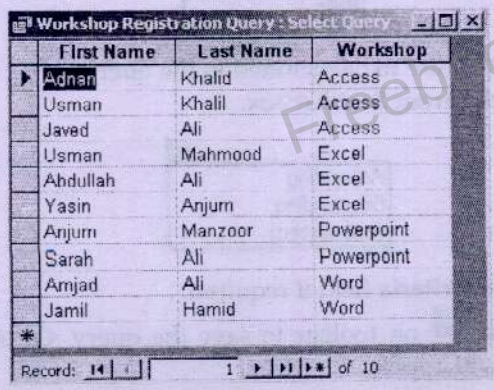

6. Use the single right arrow from **Available Fields** list box to add **First Name, Last Name** and **Workshop Registration** fields in the query.




7. Select **Next.**
8. Type a name for the query. The name appears in **what title do you want for your query?** text box.




9. Select **Finish**. The query will run and display the following records:



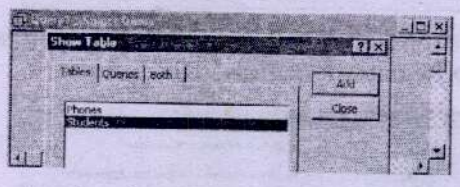

| First Name | Last Name | Workshop |
|---|---|---|
| Adnan | Khalid | Access |
| Usman | Khalil | Access |
| Javed | Ali | Access |
| Usman | Mahmood | Excel |
| Abdullah | Ali | Excel |
| Yasin | Anjum | Excel |
| Anjum | Manzoor | Powerpoint |
| Sarah | Ali | Powerpoint |
| Amjad | Ali | Word |
| Jamil | Hamid | Word |

## 13.4 Creating a Query in Design View

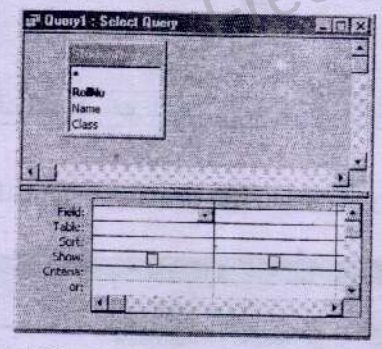
A query can be created in Design view. This option provides flexibility in designing a select query. It is used to add criteria to select records and sort the resulting RecordSet.

A design grid is used to set up the query in Design view. The field list of the table to be used in the query appears in the top pane of Design view. The user adds the fields to be used in the query to the design grid in the bottom pane of Design view along with any sort orders or criteria for selecting records. The procedure for creating simple query is as follows:

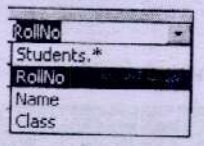
1. Open the database.
2. Click on **Queries** button in main database window.
3. Double click **Create query in design view**. **Select Query** window will appear. It will contain another dialog box **Show Table** as follows:



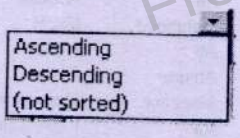

4. Select the table for extracting data.
5. Click **Add** button.
6. Click **Close** button. The **Select Query** window will appear. It contains the selected table in the upper part. The bottom part contains different options for creating query.




7. Select any field from **Field** list box to include it in the query in the bottom of window.




8. Add all other fields that are to be included in the query.
9. Select the sorting order from **Sort** list box.



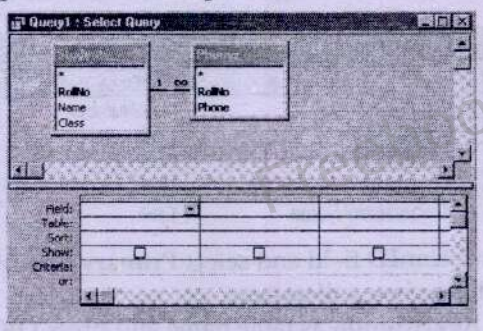

10. Give any condition in **Criteria** field if required.
11. Click on save button on toolbar to save the query. OR select **File > Save**. The **Save As** dialog box will appear.

12. Type the name of query.
13. Click **OK**. The query will be saved.
14. Select **Run** from **Query** menu to execute the query and view its result.

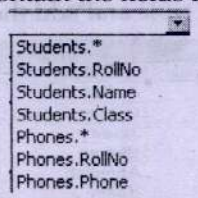
## 13.5 Creating Query from Multiple Tables in Design View

The procedure to extract data from multiple tables is as follows:

1. Open the database.
2. Click on **Queries** button in main database window.
3. Double click **Create query in design view**. **Select Query** window will appear. It will contain another dialog box **Show Table** as follows:
4. Select the first table for extracting data.
5. Click **Add** button.
6. Select the second table.
7. Click **Add** button.
8. Click **Close** button. The **Select Query** window will appear. It contains the selected table in the upper part. The bottom part contains different options for creating query.




9. In the bottom part of the window, select any field from **Field** list box to include it in the query. **Field** list box will contain the fields both tables.




10. Add all other fields that are to be included in the query.
11. Select the sorting order from **Sort** list box.
12. Give any condition in **Criteria** field if required.


13. To save the query, click on save button on toolbar. OR select **Save** from **File** menu. The **Save As** dialog box will appear.
14. Type the name of query.
15. Click **OK**. The query will be saved.
16. Select **Run** from **Query** menu to execute the query and view its result.

## Practical 4

a. Create a table called **Workshop Registration**.

Workshop Registration : Table

| Field Name | Data Type |
|---|---|
| RegistrationID | AutoNumber |
| Workshop | Text |
| FirstName | Text |
| LastName | Text |
| DepartmentName | Text |

b. Insert ten records in the table.

c. Create a query in Design view that shows First Name, Last Name and Workshop for workshop participants.

## Procedure

### a. Creating a Table

1. Create a new database in **MS Access**.
2. Double click **Create table in design view** and create the following:

Workshop Registration : Table

| Field Name | Data Type |
|---|---|
| RegistrationID | AutoNumber |
| Workshop | Text |
| FirstName | Text |
| LastName | Text |
| DepartmentName | Text |

3. Right click **RegistrationID** field and select **Primary Key** from popup menu.
4. Click **Ctrl+S**. A dialog box will appear.
5. Type **Workshop Registration** as table name and press **OK**. The table will be saved.

### b. Adding Records

1. Switch to **Datasheet View** of table and add the following records:

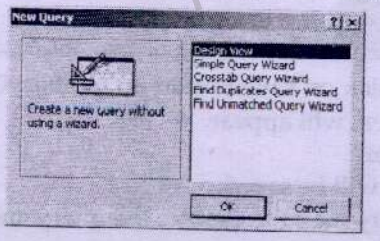
Workshop Registration : Table

| RegistrationID | Workshop | First Name | Last Name | Dept Name |
|---|---|---|---|---|
| 1 | Access | Adnan | Khalid | Economics |
| 2 | Access | Usman | Khalid | Psychology |
| 3 | Access | Javed | Ali | Psychology |
| 4 | Excel | Usman | Mahmood | Education |
| 5 | Excel | Abdullah | Ali | Education |
| 6 | Excel | Yasin | Anjum | Education |
| 7 | Powerpoint | Anjum | Manzoor | ISAT |
| 8 | Powerpoint | Sarah | Ali | ISAT |
| 9 | Word | Amjad | Ali | FM |
| 10 | Word | Jamil | Hamid | FM |
| (AutoNumber) | | | | |

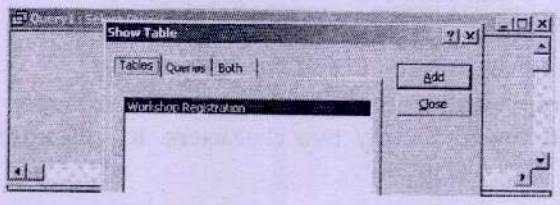
Record: 10 of 10

### c. Creating a Query

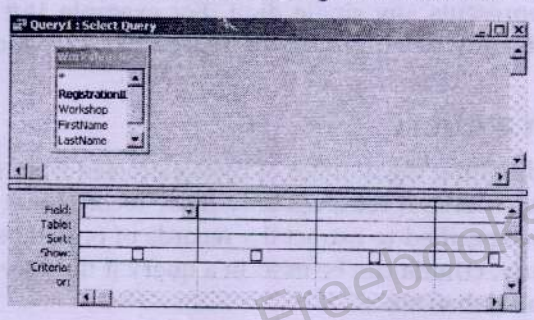
1. Click **New** button on Database window toolbar New. New Query dialog box opens.



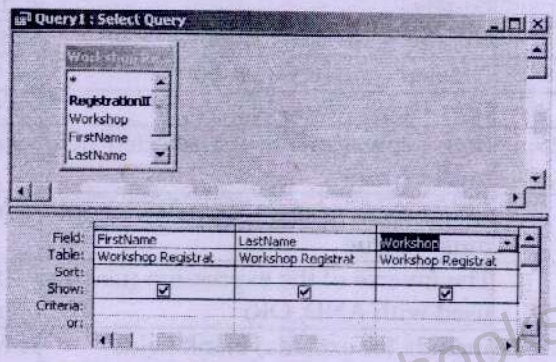
2. Select **Design View**. **Design View** is selected and a description of **Design** view appears in the left panel of the New Query dialog box.
3. Select **OK**. The New Query dialog box closes, *Design* view appears, and the Show Table dialog box opens.
4. Select a table and add it to the query. Double Click **Workshop Registration**.




5. Select **Close** to close the **Show Table dialog** box. The **Show Table** dialog box closes.




6. Select the first field you want to add to the query. Select the **First Name** field.
7. Drag the field to the desired column in the **Field** row in the design grid. A small box indicating the position of the field moves as you drag and then the field and table names appear in the design grid.
8. Add other fields to query as desired. Add ***Last* Name** and **Workshop** to the query.




9. Click the **Save** button. The Save As dialog box opens with insertion point in **Query Name** text box.
10. Type a name for the query. The name appears in the ***Query Name*** text box.
11. Select **OK**.The Save *As* dialog box closes and the query *is* saved.

## 13.6 Wild Cards

Wildcard is a special symbol that is used in queries to search data.

Some wildcards used for specifying criteria are as follows:

*

This wildcard represents any number of characters. For example "A*" indicates any text that starts with "A". It will search "ALI", "ABDULLAH" etc. "*A*" will search any text that contains "A" such as "ALI", "FAISAL" and "SEEMA" etc.

?

This wildcard represents any single character. For example "A??" indicates any text that starts with "A" followed by any two characters. It will search "ALI", "AIM" but not "AHSAN" or "AM".

#

This wildcard represents any single digit. For example 1## indicates any value that starts with 1 followed by any two digits. It will search 100, 123, and 156 but not 200, 1000 or 11 etc.

## 13.6.1 Criteria in a query

A condition used to limit the number of rows extracted from database is called **criteria**. If a query contains any criteria, it retrieves only those records that match with the specified criteria. Criteria may be specified to extract the records of only those students who got more than 700 marks. Similarly, criteria can be used in a query if the user wants to view only those students who live in Faisalabad etc.

### Specifying Criteria

Criteria are specified with the help of wild cards. Wildcards are special symbols that are used to extract particular records from the database.

| Operator | Meaning | Example |
|---|---|---|
| = | Equals | = "ITSERIES" |
| >, < | Greater Than, Less Than | >15, >#23-Jun-2002# |
| >= | Greater Than or Equal To | >= "Gilmore" |
| <= | Less Than or Equal To | <= 10 |
| <expr1> AND <expr2> | Both expressions in the selected field must be true | >5 AND <25 |
| <expr1> OR <expr2> | Either expression in the selected field must be true | ="Imran" OR ="Tariq" |
| NOT <expr> | The inverse of the expression (usually used with AND, OR) | ="Imran" AND NOT ="ImranSaeed" |
| NULL or NOT NULL | Include (or exclude) blank fields | Is Null<br>Is Not Null |
| IN (<expr1>, <expr2>,....) | Include values that match one of the items in the list | IN ("UK", "US", "JP")<br>IN (1, 4, 8, 16) |
| BETWEEN<expr1> AND <expr2> | Include values between <expr1> and <expr2>, inclusive | BETWEEN #1-MAR-2003# AND #31-MAR-2003# |

## Adding Criteria for a Field

- Select the **Criteria:** row for the appropriate field
- Type in the criteria by which you wish to filter the query

## Adding Criteria for Multiple Fields

You can add criteria for several fields on the **Criteria:** row - all of the criteria will have to be true for a record to be displayed in the query result.

You can add criteria to one or more of the **Or:** rows below the **Criteria:** row to create multiple sets of alternative criteria for the same field.

## Practical 5

a. Create a table Employee with the following structure (EmpNo as primary key):

| FieldName | Datatype |
|---|---|
| EmpNo | Number |
| EmpName | Text |
| Job | Text |
| Salary | Number |

b. Add given records in table:

| EmpNo | EmpName | Job | Salary |
|---|---|---|---|
| 1 | Ali | CLERK | 2000 |
| 2 | Jamil | MANAGER | 8000 |
| 3 | Jamal | CLERK | 2500 |
| 4 | Tasleem | ANALYST | 10000 |
| 5 | Imran | ANALYST | 8000 |

c. Create query in design view to find information about the employee whose EmpNo is 4.

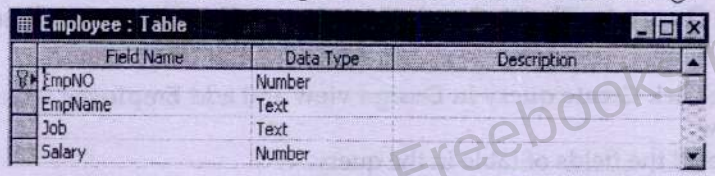
d. Create query in design view to find information about all those employee(s) not having EmpNo 3.

e. Create query to find information about employee whose name begin with "Ja".

## Procedure

### a. Creating a Table

1. Create a new database in **MS Access**.
2. Double click **Create table in design view** and create the following:




3. Right click **EmpNo** field and select **Primary Key** from popup menu.
4. Click **Ctrl+S**. A dialog box will appear.
5. Type **Employee** as table name and press **OK**. The table will be saved.

### b. Adding Records

1. Switch to **Datasheet View** of table and add the following records:

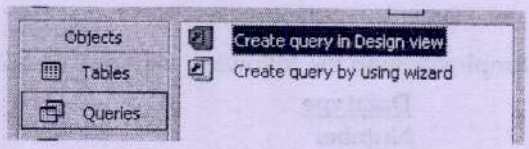
Employee : Table

| | EmpNo | EmpName | Job | Salary |
|---|---|---|---|---|
| | 1 | Ali | CLERK | 2,000 |
| | 2 | Jamil | MANAGER | 8,000 |
| | 3 | Javed | CLERK | 2,500 |
| | 4 | Tasleem | ANALYST | 10,000 |
| | 5 | Imran | ANALYST | 80,000 |

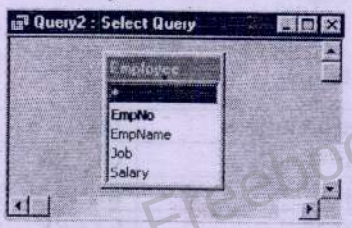
Record: 5 of 5

### c. Creating a Query to Find Information about Employee whose EmpNo is 4

1. Select **Queries** option in **Objects** list of **Database window** and double click **Create query in Design view.**




2. Add **Employee** table to the **Query** window from **Show Table** dialog box.




3. Choose all the fields of table in the query.
4. Type **4** in **Criteria** expression of **EmpNo** field.

| Field: | EmpNo | EmpName | Job | Salary |
|---|---|---|---|---|
| Table: | Employee | Employee | Employee | Employee |
| Sort: | | | | |
| Show: | ☑ | ☑ | ☑ | ☑ |
| Criteria: | 4 | | | |
| or: | | | | |

5. Switch to **Datasheet View** of query to see the result.

Query2 : Select Query

| | EmpNo | EmpName | Job | Salary |
|---|---|---|---|---|
| ▶ | 4 | Tasleem | ANALYST | 10,000 |
| * | 0 | | | 0 |

### d. Creating a Query to Find Information about all those Employees not having Empno 3

1. Double click **Create query in Design view** and add **Employee** table to **Query** window.
2. Choose all the fields of table in the query.
3. Type **<>3** in **Criteria** box under **EmpNo** field.

| Field: | EmpNo | EmpName | Job | Salary |
|---|---|---|---|---|
| Table: | Employee | Employee | Employee | Employee |
| Sort: | | | | |
| Show: | ☑ | ☑ | ☑ | ☑ |
| Criteria: | <>3 | | | |
| or: | | | | |

4. Switch to **Datasheet View** of query to see the result.

Query3 : Select Query

| EmpNo | EmpName | Job | Salary |
|---|---|---|---|
| 1 | Ali | CLERK | 2,000 |
| 2 | Jamil | MANAGER | 8,000 |
| 4 | Tasleem | ANALYST | 10,000 |
| 5 | Imran | ANALYST | 80,000 |

Record: 5 of 5

### e. Creating Query to Find Information of all those Employees whose name Begin with "Ja"

1. Select **Create query in Design view** and add **Employee** table to the **Query** window.
2. Choose all the fields of table in the query.
3. Type **Like "Ja*"** in **Criteria** box under **EmpName** field.

| Field: | EmpNo | EmpName | Job | Salary |
|---|---|---|---|---|
| Table: | Employee | Employee | Employee | Employee |
| Sort: | | | | |
| Show: | ☑ | ☑ | ☑ | ☑ |
| Criteria: | | Like "Ja*" | | |

4. Switch to **Datasheet View** of query to see the result.

Query4 : Select Query

| EmpNo | EmpName | Job | Salary |
|---|---|---|---|
| 2 | Jamil | MANAGER | 8,000 |
| 3 | Jamal | CLERK | 2,500 |

## Practical 6

a. Create a table called **Client** as follows:

| **Fieldname** | **Data type** |
|---|---|
| Salesman No | Text |
| Name | Text |
| Sale | Number |
| City | Text |

b. Enter the following records in the table:

| Salesman No | Name | Sale | City |
|---|---|---|---|
| S001 | Aslam | 5000 | FSD |
| S002 | Ali | 2000 | FSD |
| S003 | Ejaz | 3000 | LHR |
| S004 | Nadeem | 7000 | LHR |
| S005 | Naeem | 9000 | KHI |

c. Create query in design view to display the entire salesmen who are located in FSD.
d. Create query in design view to find out the salesman who stays in a city whose second letter is 'H'.

## Procedure

### a. Creating a Client Table

1. Create a table in design view.
2. Select **SalesmanNo** as **Primary Key**.

Client : Table

| Field Name | Data Type | Description |
|---|---|---|
| SalesmanNo | Text | Identification number |
| Name | Text | Name |
| Sale | Number | Income |
| City | Text | Residance |

3. Save the table as **Client**.

## b. Adding Records

4. Switch to **Datasheet View** of table and enter the following records:

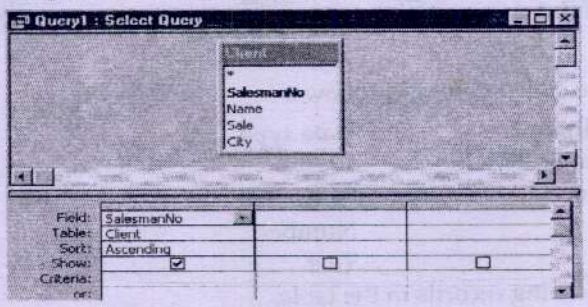
Client : Table

| SalesmanNo | Name | Sale | City |
|---|---|---|---|
| S001 | Aslam | 5000 | FSD |
| S002 | Ali | 2000 | FSD |
| S003 | Ejaz | 3000 | LHR |
| S004 | Nadeem | 7000 | LHR |
| S005 | Naeem | 2000 | KHI |

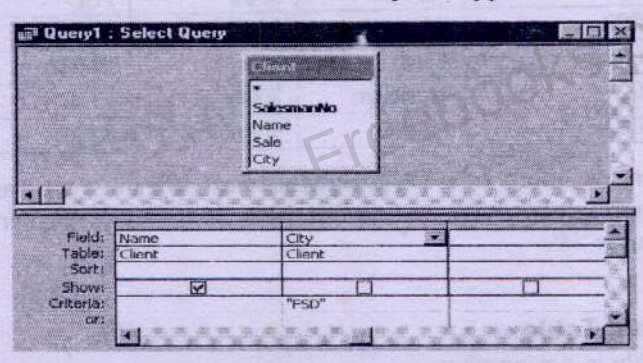
Record: 1 of 5

## c. Creating a Query to Display All Salesmen located in FSD

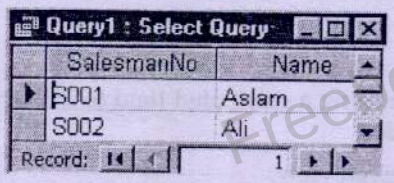
1. Double click **Create query in Design view** from **Queries** tab in **Database** window.
2. Add **Client** from **Show Table** dialog box.
3. Click **Close** button to close the dialog box.
4. Click in first cell of **Field.**
5. Choose **SalesmanNo** from drop down arrow on right of cell.
6. Choose **Ascending** from **Sort** option and select **Show** checkbox.




7. Select **Name** in second **Field** and check the **Show** box.
8. Uncheck **Show** box in third **Field** named **City** and type "**FSD**" in **Criteria** box.

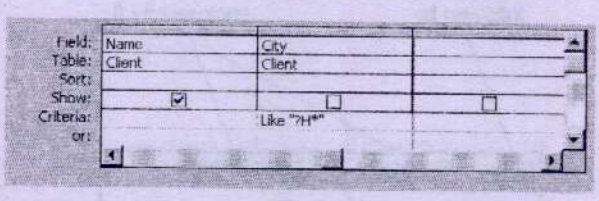



9. Switch to **Datasheet View** of query to see the result.




**d. Creating a Query to find out the Salesman who Stays in a City whose Second Letter is 'H'**

1. Create another query as above.
2. Type **Like "?H*"** instead of **"FSD"** in **Criteria** under **City** field.




3. Switch to **Datasheet View** of query to see the result.

| SalesmanNo | Name |
|---|---|
| S003 | Ejaz |
| S004 | Nadeem |
| S005 | Naeem |

# 13.7 Adding a Calculated Field to a Query

Some fields are calculated instead of storing in a table. For example, if a field **Date of Birth** field exists in a table then **Age** field is not required. The age can be calculated by using **Date of Birth** field. Some examples of calculated fields are as follows:

| This Expression | Displays |
|---|---|
| [UnitPrice] * [Quantity] | The results of the multiplication of these two fields |
| [OrderDate] + 28 | The value of the OrderDate field plus four weeks (28 days) |
| [FirstName] + [MiddleInitial] + [LastName] | The full name from three fields that contain name information |
| [TotalScore] / [NoOfQuestions] | The average score per question |

## 13.7.1 Expression Syntax

The following format is used to create a calculated expression to display in a query:

***FieldName:*** *<expression>*

### Example

ItemValue: [UnitPrice] * [Quantity]

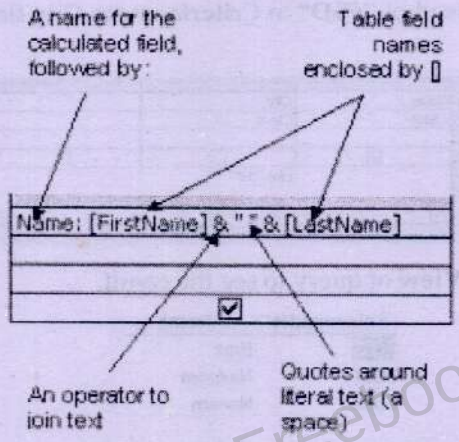
The field name should be enclosed with square brackets [ ]. It is compulsory if the field contains spaces. MS Access shows a syntax error if brackets are not used. If a field name occurs twice in the data source then the user must identify the object that contains it.

[Table]. [Field]

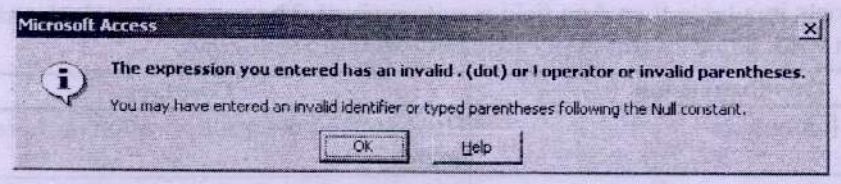
### Writing an Expression

The simplest method of adding a calculated field to a query is to type the expression in the **Field:** cell of the query builder.

- Display the query in design view.
- Click **Field:** cell into which you will type the expression and type the expression
- Press or move off the field.




If you have made a mistake in the structure of the calculated field, Access will report a syntax error. For example:




*Syntax error dialogue box*

- Edit the expression until it passes the syntax checking
- Make sure that you spell the field names correctly

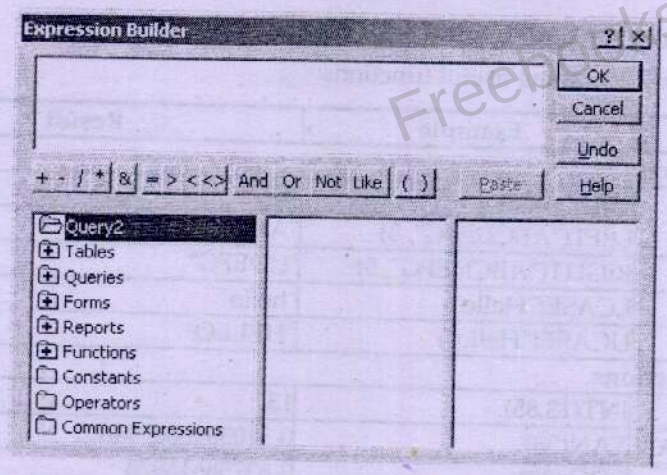
### Editing an Expression

- Select the field that contains the expression. You will find that the entire field is selected, and you cannot edit the field without first deleting the contents.
- Press to enable editing of the expression. OR
- Press + to zoom into a complex expression

## 13.8 Expression Builder

A simple expression can be typed quickly and easily. The more complex expressions are hard to create and edit in such a small area. A simple typing mistake can be hard to find. MS Access provides an **Expression Builder** that provides facility to create and edit complex expressions using a visual tool. The following method is used to enter a calculated field using Expression Builder:

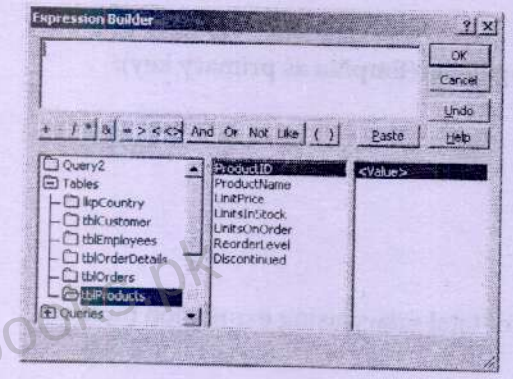
- Select a blank field
- Click **Build** on **Standard** toolbar. **Expression Builder** dialog box will appear:

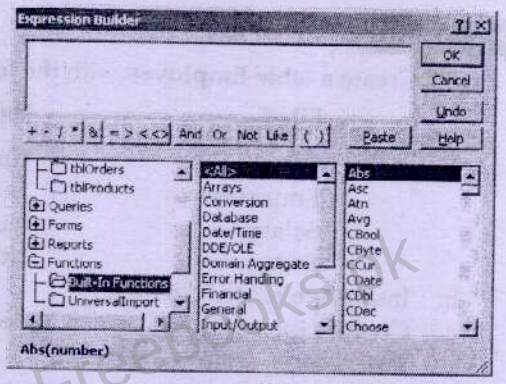



The dialog box is divided in three sections. The **top** section is used to build expression. The **middle** section has a row of buttons to access all standard expression operators quickly. The **bottom** section contains three list boxes that provide various values to be selected and pasted in the top textbox. The left list box contains the groupings of available objects or functions. If a group folder is marked with a plus symbol ⊞ it contains subfolders.

- Double-click a group folder ⊞ to expand the list of available subfolders
- Double-click an expanded group folder ⊟ to collapse it again
- Click a bottom level folder 🗀 to display its contents in the other two list boxes

If the user selects a folder containing objects from the database, a list of objects and the properties of that object will appear. A list of all built-in functions appears if the user opens **Functions** folder.




The centre text box displays a list of function groups. It is used to find the required function. The user can select appropriate function or value and then move it in the textbox at the top of the dialog box.

- Click **Paste** to include the selected function or value in the expression
- Optionally, type in the text box to edit the expression

**Note:** **Paste** uses the selected value from the right-hand list box.

- When the expression is complete, click **OK**

## Built-in Functions for Calculated Field Expressions

The table below lists some useful functions:

| | Example | Result |
|---|---|---|
| **String Functions** | | |
| MID | MID("ABCDEFG", 4,2) | "DE" |
| LEFT | LEFT("ABCDEFG", 3) | "ABC" |
| RIGHT | RIGHT("ABCDEFG", 5) | "CDEFG" |
| LCASE | LCASE("Hello") | "hello" |
| UCASE | UCASE("Hello") | "HELLO" |
| **Number Functions** | | |
| INT | INT(13.85) | 13 |
| SIN, COS, TAN | TAN(38) | 0.310309660994801 |
| LOG | LOG (38) | 3.63758615973 |
| ABS | ABS(-15) | 15 |
| SQR | SQR(64) | 8 |
| **Date/Time Functions** | | |
| NOW | NOW() | Returns the current date and time |
| DATEDIFF | DATEDIFF("yyyy", [DateOfBirth], NOW()) | Calculates the age in years of an employee from their birth date and today (NOW()) |
| DATEPART | DATEPART("m", NOW()) | Gives the current number of the month - for example, if you are in July the function returns 7 |
| DATEADD | DATEADD("q", 3, NOW()) | Returns the date three quarters (or 9 months) from today |

## Practical 7

a. Create a table **Employee** with the following fields (**EmpNo** as primary key):

| Filed | Datatype |
|---|---|
| EmpNo | Number |
| Empname | Text |
| Salary | Number |
| Bonus | Number |

b. Insert five records in table.

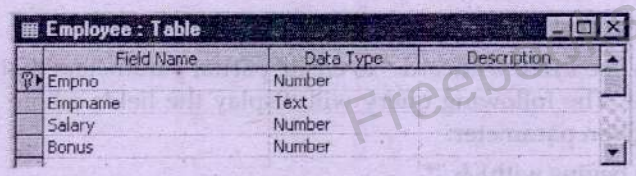
c. Create a query to display calculated field of total salary using expression builder: Totalsalary = salary + bonus.

d. Create a report using autoreport.

## Procedure

### a. Creating a Employee Table

1. Create a table in design view.
2. Select **Empno** as primary key.



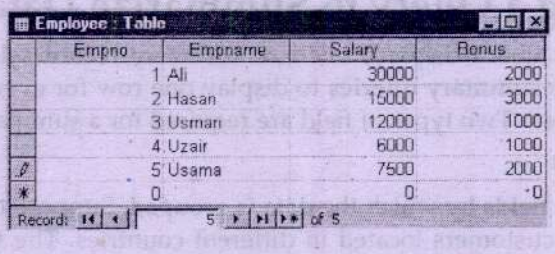
Employee : Table

| Field Name | Data Type | Description |
|---|---|---|
| Empno | Number | |
| Empname | Text | |
| Salary | Number | |
| Bonus | Number | |

3. Save the table as Employee.

### b. Adding Records

1. Switch to **Datasheet View** of table and enter the following records:



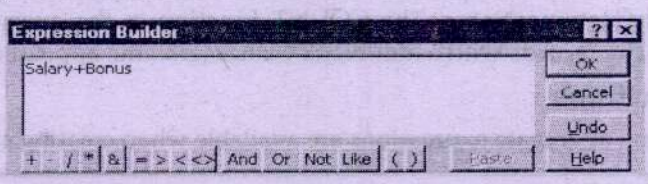
Employee : Table

| Empno | Empname | Salary | Bonus |
|---|---|---|---|
| 1 | Ali | 30000 | 2000 |
| 2 | Hasan | 15000 | 3000 |
| 3 | Usman | 12000 | 1000 |
| 4 | Uzair | 6000 | 1000 |
| 5 | Usama | 7500 | 2000 |
| 0 | | 0 | 0 |

Record: 5 of 5

### c. Creating a Query

1. Double click **Create query in Design view** from **Queries** tab in **Database** window.
2. Add **Employee** table to **Query** window.
3. Choose **Empname** and **Salary** fields in the query.
4. Right click **Salary** field and select **Build** from pop up menu. The **Expression Builder** will appear.
5. Type **Salary+Bonus** in the text box and click **OK**.



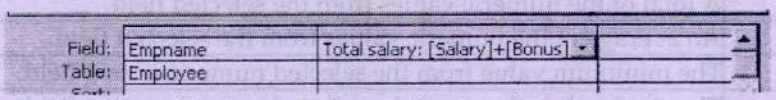
Expression Builder

Salary+Bonus

OK
Cancel
Undo
Help

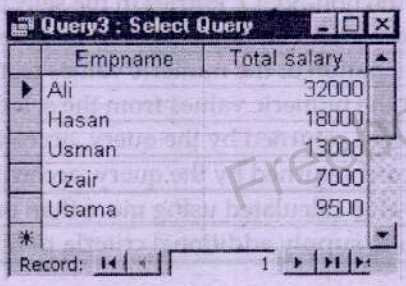
+ - / * & = > < <> And Or Not Like ( ) Paste

6. Type **Total salary** in place of **Expr1**. The query will appear as follows:



| | | |
|---|---|---|
| Field: | Empname | Total salary: [Salary]+[Bonus] |
| Table: | Employee | |

7. Switch to **Datasheet View** to see the results.



Query3 : Select Query

| Empname | Total salary |
|---|---|
| Ali | 32000 |
| Hasan | 18000 |
| Usman | 13000 |
| Uzair | 7000 |
| Usama | 9500 |

Record: 1

## 13.9 Creating a Parameter Query

A **parameter query** displays a dialog box when it is executed. It gets some information from the user as parameter. These queries are used as basis for creating forms and reports.

### Wildcards in Parameters

The user can use **LIKE** wildcard * to create partial parameters that will work on part of the associated field. The following query will display the fields where the beginning of the field matches the given parameter:

LIKE [Name begins with] & "*"

The following query will return fields where the parameter is contained in the field:

LIKE "*" [Name contains] & "*"

## 13.10 Creating a Query to Summarize Data

A simple select query displays one row for every record selected by the query. The user can also produce **summary queries** to display one row for every change in data for the fields to be summarized. Two types of field are required for a summary query:

### Group by Field(s)

It is the field or fields by which the data is grouped. Suppose the user needs a query to show the number of customers located in different countries. The **Group By** field in query will be **Country**. The query results will display one line for each country. The grouping field does not need to be a validated field such as a foreign key field. However, the field must contain duplicate information by which the query result is grouped. The user will get one summarized value based on all records selected by query if no grouping fields are included.

### Aggregate Fields

These are the fields that provide the values to calculate. The aggregate field in the above example is **CustomerName**. The COUNT **function** in the query groups the data by country. It counts the records with a non-blank CustomerName appeared in each group.

### Aggregate Functions

The following aggregate expressions are available when creating summary queries:

| Expression | Result |
|---|---|
| SUM | A total of the numeric values from the selected field. |
| AVG | An average of the numeric values from the selected field. |
| MIN | The minimum value from the selected numeric or text field. |
| MAX | The minimum value from the selected numeric or text field. |
| COUNT | A count of the non-NULL entries in the selected field.<br>If you want to count all of the records returned by a query, use COUNT(*). |
| STDEV | Standard deviation of the numeric values from the selected field. |
| VAR | Variance of the numeric values from the selected field. |
| FIRST | The first record returned by the query, according to the query SORT order. |
| LAST | The last record returned by the query, according to the query SORT order. |
| EXPRESSION | Returns a value calculated using more than one summary function. |
| WHERE | Allows you to supply additional criteria to filter the expression. |

These functions only work with **Number, Date/Time, and Currency** and **AutoNumber** data types. **Count** function work with most data types. The user can also use calculated field as aggregate field. Suppose a calculated field is created to show OrderValue. A summary query can be created to show total value of orders from different customers. The group by field will be CustomerName and aggregate field will be OrderValue.

The SUM function will be used to add the total of orders for each customer. In this case, the user must enter EXPRESSION as aggregate function. The user can group results by more than one field by adding another **Group By** totals field to query. The order of grouping depends on the order of **Group By** fields in query.

## Practical 8

a. Create the following tables in Access:
**Employee** (EmpId, EmpName, Job, Sal, DeptNo)
**Department** (DeptNo, Dname, Loc)

b. Develop relationship between both the tables.

c. Insert the following records in **Department** table:

| DeptNo | Dname | Loc |
|---|---|---|
| 10 | Accounting | Faisalabad |
| 20 | Research | Karachi |
| 30 | Sales | Multan |
| 40 | Operations | Lahore |

d. Insert the following records in **Employee** table:

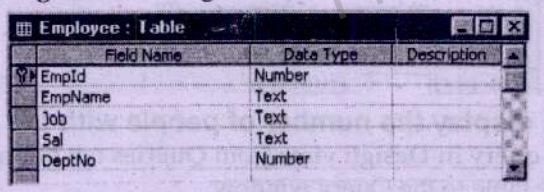
| EmpNo | EmpName | Job | Sal | DeptNo |
|---|---|---|---|---|
| 1 | Salman | Clerk | 800 | 20 |
| 2 | Ali | Salesman | 1200 | 30 |
| 3 | Wahid | Salesman | 1250 | 30 |
| 4 | Jamil | Manager | 2975 | 20 |
| 5 | Mahmood | Salesman | 1230 | 30 |
| 6 | Bilal | Manager | 3000 | 30 |
| 7 | Amar | Manager | 2800 | 10 |
| 8 | Saleem | Analyst | 5000 | 20 |
| 9 | Ahmed | Clerk | 500 | 20 |
| 10 | Tauseef | Salesman | 1000 | 30 |

e. Write a query to display the number of people with same job.
f. Write a query to display the minimum salary for each job type.
g. Write query to display all departments which have more than 3 employees
h. Write a query to display the minimum salary earned by clerk.
i. Write a query to calculate the average salary of all employees.
j. Write a query to list the employee names and salary increased by 15%.

## Procedure

### a. Creating a Employee Table

1. Create the following table in design view.




2. **Select EmpId as Primary Key.**
3. **Save the table as Employee.**

### Creating a Department Table

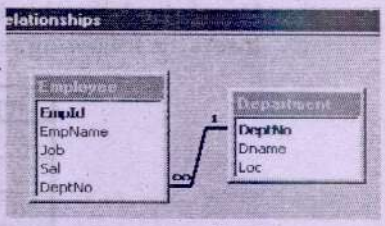
1. Create the following table in design view.

Department : Table

| Field Name | Data Type | Description |
|---|---|---|
| DeptNo | Number | |
| Dname | Text | |
| Loc | Text | |

2. Select **DeptNo** as **Primary Key.**
3. Save the table as **Department.**

### b. Developing Relationship between Table




1. Click **Tools > Relationship**. A dialog box will appear.
2. Select both tables one by one and click **Add** button.
3. Click **Close** to close the dialog box.
4. Click on **DeptNo** in **Department** table, drag it on **DeptNo** in **Employee** table and release the mouse. A dialog box will appear.
5. Select **Enforce Referential Integrity** checkbox and click **Create** button. Both tables will be joined by one-to-may relationship as follows.
6. Save the changes and close **Relationship** window.

### c. Adding Records in Department Table

1. Open **Department** table in **Datasheet View** and add the following records.

Department : Table

| DeptNo | Dname | Loc |
|---|---|---|
| 10 | Accounting | FSD |
| 20 | Research | KHI |
| 30 | Sales | Multan |
| 40 | Operations | LHR |
| 0 | | |

Record: 4 of 4

### d. Adding records in Employee Table

1. Open **Employee** table in **Datasheet View** and add the following records.

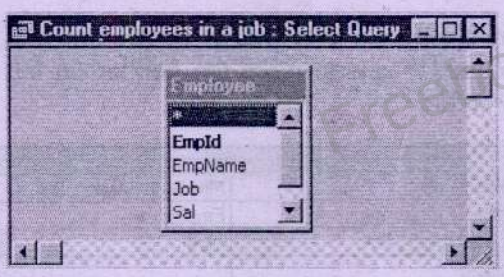
Employee : Table

| EmpId | EmpName | Job | Sal | DeptNo |
|---|---|---|---|---|
| 1 | Salman | Clerk | 800 | 20 |
| 2 | Ali | Salesman | 1200 | 30 |
| 3 | Wahid | Salesman | 1250 | 30 |
| 4 | Jamil | Manager | 2975 | 20 |
| 5 | Mahmood | Salesman | 1230 | 30 |
| 6 | Bilal | Manager | 3000 | 30 |
| 7 | Amar | Manager | 2800 | 10 |
| 8 | Saleem | Analyst | 5000 | 20 |
| 9 | Ahmed | Clerk | 600 | 20 |
| 10 | Tauseef | Salesman | 1000 | 30 |
| 11 | Jamal | Clerk | 850 | 30 |
| 12 | Faheem | Analyst | 4500 | 20 |

Record: 12 of 12

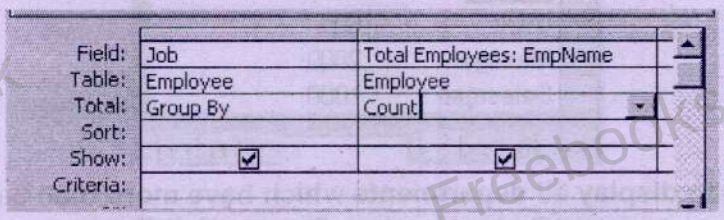
### e. Write a query to display the number of people with the same Job

1. Select **Create query in Design view** from **Queries** tab in Database window.
2. Add *Employee* table to the Query window.



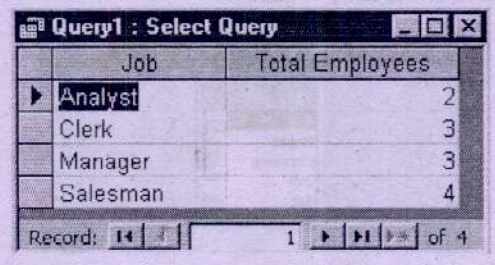

3. Choose *Job* & *EmpName* fields in the query.

3. Now to display summary criteria, from the **Query Design** toolbar, click **Totals** Σ
4. An additional **Total:** row is added to the query grid.
5. Click the **Total:** row for the *EmpName* field and select **Count** function.
6. To rename *EmpName* column in query, place the pointer on left of *EmpName* & then type *Total Employees* followed by colon.



| | | |
|---|---|---|
| Field: | Job | Total Employees: EmpName |
| Table: | Employee | Employee |
| Total: | Group By | Count |
| Sort: | | |
| Show: | ☑ | ☑ |
| Criteria: | | |

7. Now switch to **Datasheet view** of query to see the results.



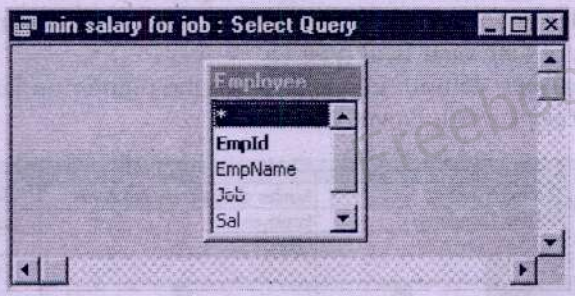
Query1 : Select Query

| Job | Total Employees |
|---|---|
| Analyst | 2 |
| Clerk | 3 |
| Manager | 3 |
| Salesman | 4 |

Record: 1 of 4

### f. Write a query to display the minimum salary for each job type

1. Select **Create query in Design view** from **Queries** tab in **Database** window.
2. Add **Employee** table to **Query** window.




3. Choose *Job* & *Sal* fields in the query.

4. Now to display summary criteria, from the **Query Design** toolbar, click **Totals** Σ
5. An additional **Total:** row is added to the query grid.

6. Click the **Total:** row for the *Sal* field and select **Min** function.
7. To rename *Sal* column in query, place the pointer on left of *Sal* & then type *Min Salary* followed by colon.

| | | |
|---|---|---|
| Field: | Job | Minimum Salary: Sal |
| Table: | Employee | Employee |
| Total: | Group By | Min |
| Sort: | | |
| Show: | ☑ | ☑ |
| Criteria: | | |

8. Now switch to **Datasheet view** of query to see the results.

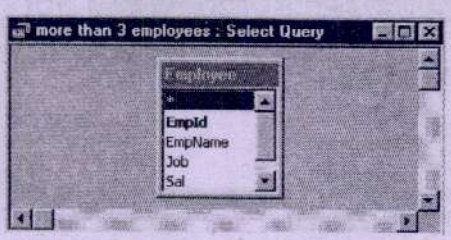
min salary for job : Select ...

| Job | Minimum Salary |
|---|---|
| Analyst | 4500 |
| Clerk | 500 |
| Manager | 2800 |
| Salesman | 1000 |

Record: 1

## g. Write query to display all departments which have more than 3 employees

1. Select **Create query in Design view** from **Queries** tab in Database window.
2. Add *Employee* table to the Query window.




3. Choose *DeptNo* & *EmpName* fields in the query.
4. Now to display summary criteria, from the **Query Design** toolbar, click **Totals** Σ
5. An additional **Total:** row is added to the query grid.
6. Click the **Total:** row for the *EmpName* field and select **Count** function.
7. In **Criteria** box of *EmpName* field type >3.
8. To rename *EmpName* column in query, place the pointer on left of *EmpName* & then type *Number of employees* followed by colon.

| | | |
|---|---|---|
| Field: | DeptNo | Number of employees: EmpName |
| Table: | Employee | Employee |
| Total: | Group By | Count |
| Sort: | | |
| Show: | ☑ | ☑ |
| Criteria: | | >3 |
| or: | | |

9. Now switch to **Datasheet view** of query to see the results.

more than 3 employees : Select ...

| DeptNo | Number of employees |
|---|---|
| 20 | 5 |
| 30 | 6 |

Record: 1 of 2

## h. Creating a Query to Display the Minimum Salary Earned by Clerk

1. Double click **Create query in Design view** from **Queries** tab in **Database** window.
2. Add **Employee** table to **Query** window.
3. Choose **Job** and **Sal** fields in the query.
4. Type **Like "Clerk"** in **Criteria** box of **Job** field.
5. Click **Totals** Σ from **Query Design** toolbar to display summary criteria. A **Total:** row will be added to the query grid.
6. Click **Total:** row for **Sal** field and select **Min** function.
7. Type **Min salary:** before **Sal** to rename **Sal** column in query.

| | | |
|---|---|---|
| Field: | Job | Min salary: Sal |
| Table: | Employee | Employee |
| Total: | Group By | Min |
| Sort: | | |
| Show: | ☑ | ☑ |
| Criteria: | "Clerk" | |
| or: | | |

8. Switch to **Datasheet view** of query to see the results.

min salary of clerk : Selec...

| Job | Min salary |
|---|---|
| Clerk | 500 |

9. Save the query and close it.

## i. Creating a Query to Display the Average Salary of all Employees

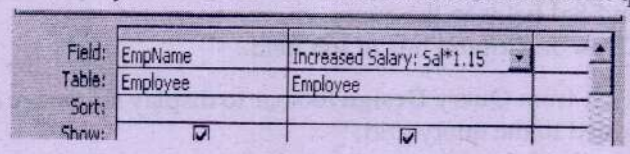
1. Double click **Create query in Design view** from **Queries** tab in **Database** window.
2. Add **Employee** table to **Query** window.
3. Choose **Sal** field in the query.
4. Click **Totals** Σ from **Query Design** toolbar to display summary criteria. A **Total:** row will be added to the query grid.
5. Click **Total:** row for **Sal** field and select **Avg** function.
6. Type **Average salary:** before **Sal** to rename **Sal** column in query.

| | | |
|---|---|---|
| Field: | Average Salary: Sal | |
| Table: | Employee | |
| Total: | Avg | |
| Sort: | | |
| Show: | ☑ | ☐ |
| Criteria: | | |

7. Switch to **Datasheet view** of query to see the results.

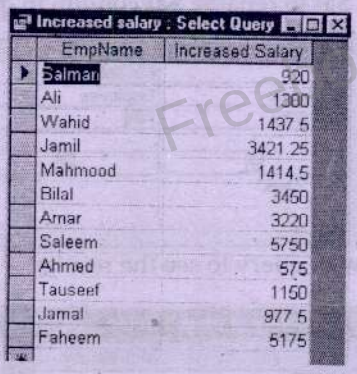

### j. Creating a Query to List the Employee Names and Salary Increased by 15%

1. Double click **Create query in Design view** from **Queries** tab in **Database** window.
2. Add **Employee** table to **Query** window.
3. Select **EmpName** and **Sal** fields in the query.
4. Place pointer in **Sal** field and type ***1.15** on right.
5. Type **Increased salary:** before **Sal*1.15** to rename **Sal** column in query.



| Field: | EmpName | Increased Salary: Sal*1.15 |
|---|---|---|
| Table: | Employee | Employee |
| Sort: | | |
| Show: | ☑ | ☑ |

6. Switch to **Datasheet view** of query to see the results.



Increased salary : Select Query

| EmpName | Increased Salary |
|---|---|
| Salman | 920 |
| Ali | 1300 |
| Wahid | 1437.5 |
| Jamil | 3421.25 |
| Mahmood | 1414.5 |
| Bilal | 3450 |
| Amar | 3220 |
| Saleem | 5750 |
| Ahmed | 575 |
| Tauseef | 1150 |
| Jamal | 977.5 |
| Faheem | 5175 |

## 13.11 Action Query

An **action query** is used to make changes in specified records on an existing table. It is also used to create a new table. There are four types of action queries:

- **Delete Query:** It is used to delete a group of records from one or more tables.
- **Update Query:** It is used to make changes to a group of records in one or more tables.
- **Append Query:** It is used to add a group of records from one or more tables to the end of one or more tables.
- **Make Table Query:** It is used to create a new table and copy selected records in it.

### 13.11.1 Make-Table Query

A make-table query creates new table from all or part of the data in one or more tables. This query is helpful for:

- Making a backup copy of a table.
- Creating a table to export to other Access databases.
- Creating a history table that contains old records.

## Practical 9

Create a table called client

| Field Name | Data Type |
|---|---|
| ClientNo | Number |
| Name | Text |
| Address | Text |
| City | Text |

### Perform the following tasks:

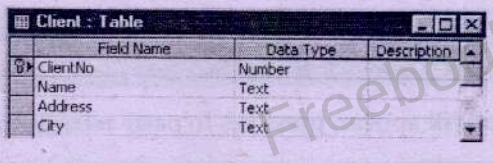
a) Add five records to the table.

b) Use make-table query to create table *Supplier from Client*. Select all the clients whose name begin with U or A; rename ClientNo with SupplierNo and Name with SupplierName.

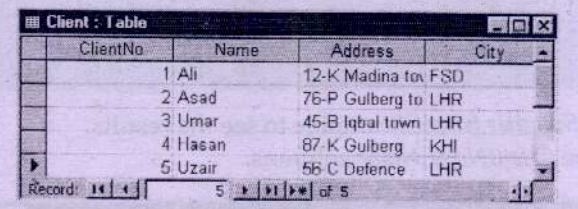
### Procedure

### a. Create 'Client' and add records

1. Create a table in design view.
2. Select ClientNo as **Primary Key**.

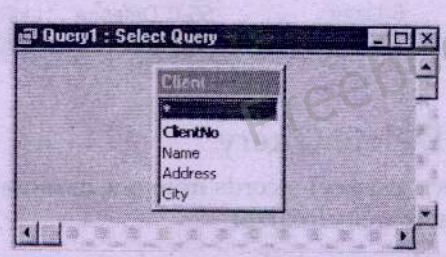



3. Save the table as *Client*.
4. Switch to **Datasheet** View of table & enter data as mentioned.

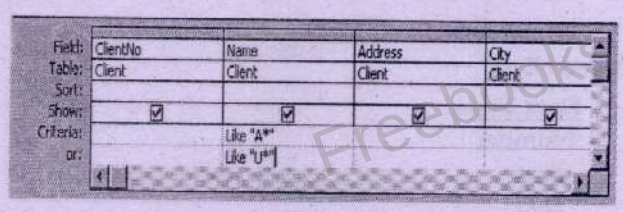



### b. Create Make table query

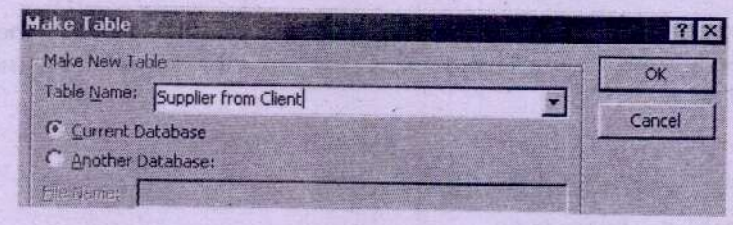
1. Select Create query in Design view from Queries tab in Database window.
2. Add *Client* table to the Query window.




3. Choose all the fields of table in the query.
4. In **Criteria** expression of *Name* **Field** type *Like "A*"*.
5. In **OR** option of *Name* **Field** type *Like "U*"*.

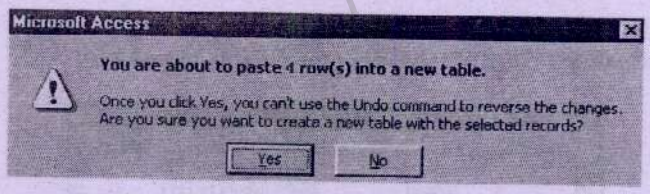



6. Now choose **Make-Table Query** option from **Query Type** tool.
7. **Make Table** dialog box will appear.
8. Type *Supplier from Client* in **Table Name** text box and press OK.

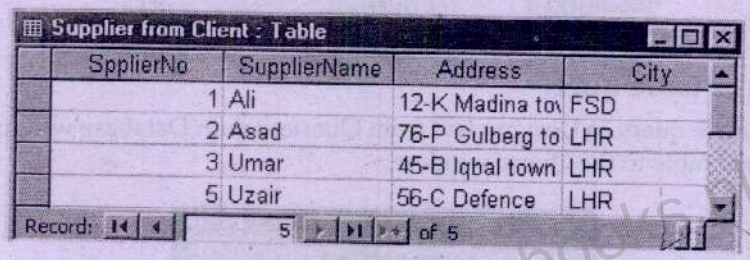



9. Now press **Run** tool to action the query,


10. A confirmation box will appear, press yes to paste records in newly formed table.




11. Now open the *Supplier from Client* table to see the results.
12. Also rename the *ClientNo* & *Name* columns.



Supplier from Client : Table

| SpplierNo | SupplierName | Address | City |
|---|---|---|---|
| 1 | Ali | 12-K Madina to | FSD |
| 2 | Asad | 76-P Gulberg to | LHR |
| 3 | Umar | 45-B Iqbal town | LHR |
| 5 | Uzair | 56-C Defence | LHR |

Record: 5 of 5

## 13.11.2 Append and Delete Query

An **append query** adds a group of records from one or more tables to the end of one or more tables. The append queries are helpful for:

- Appending fields based on criteria. For example, the user can append only the names and addresses of customers with outstanding orders.
- Appending records when some of the fields in one table do not exist in other table.

## Delete Query

A **delete query** deletes a record or group of records from one or more tables. The user cannot undo the action of deleting the record using a delete query.

**Practical 10**

Create a table called Student:

| Field Name | Data Type |
|---|---|
| Rollno | Number |
| Studentname | Text |
| Classname | Text |

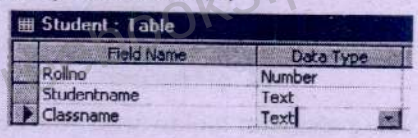
### Perform the following tasks:

a) Insert at-least five records in table.
b) Use append-query to display Rollno, Studentname, Classname for MSCS class to a table named Student.
c) Use Delete query to delete all records having Classname MSCS, from Student table

**Procedure**

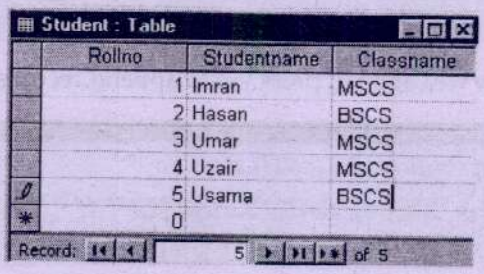
### Create 'Student' table (Rollno, Studentname, Classname)

1. Create a table in design view.
2. Use the same fields as above mentioned.
3. Save the table as *Student*.



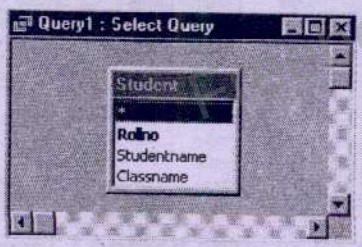

### Insert records in table

1. Switch to **Datasheet View** of table and enter data in all fields for five records:

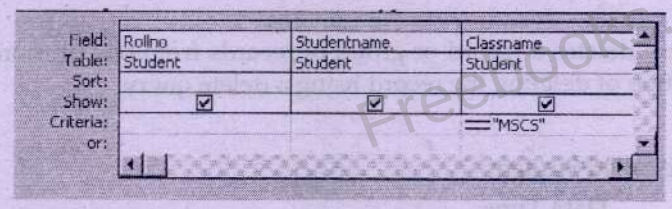



### Creating Append query

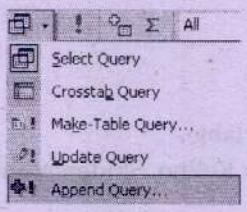
1. Select **Create query in Design view** from Queries tab in Database window.
2. Add **Student** table to **Query** window.



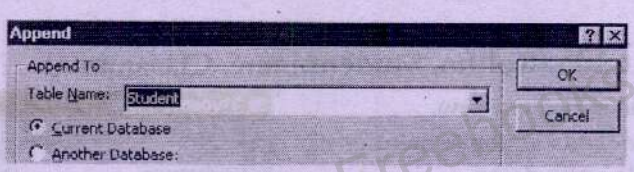

3. Choose all the fields of table in the query.
4. Type "MSCS" in **Criteria** under **Classname** field.




5. Now on the **Standard** toolbar choose **Append** Query option from Query Type tool. **Append** dialog box will appear.



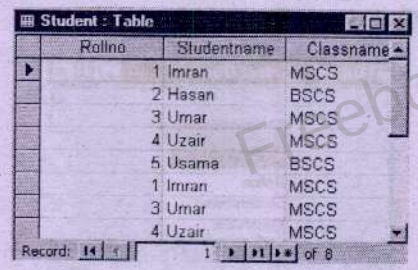

6. Select *Student* from **Table Name** drop down list and press OK.




7. Now press **Run** tool to action the query.


8. A confirmation box will appear, press yes to append records in student table.

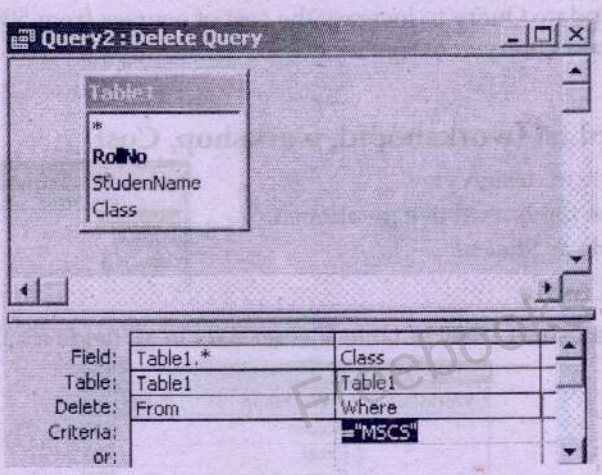

9. Now open the Student table to see the results. Result of query will be appended to Student table.



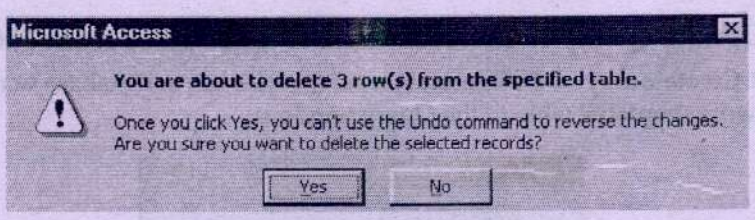

## Use Delete query to delete all records having Classname *MSCS*

1. Open the previously created Query in **Design View.**
2. Click **Query Type** on **Standard** toolbar and select **Delete Query** from list.

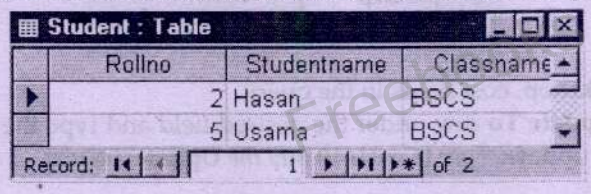

3. The **Sort** and **Append To** lines are replaced by the **Delete** line on the query grid.




4. Now to run the query, from the **Standard** toolbar, click **Run** The **Delete** message box is displayed.




5. Press **Yes** to apply the query.
6. Switch to **Datasheet View** of *Student* table to see the effect of query.




## 13.11.3 Creating an Update Query

An **Update query** is to change the values of data in an existing table. It saves time by updating a large number of records in a table at once. For example, an update query can be used to increase the values in a **Unit** Price field by 10%.

## Practical 11

Create a table called WorkshopList

| Field Name | Data Type |
|---|---|
| Workshopid | AutoNumber |
| Workshop | Text |
| Cost | Number |

### Perform the following tasks:

a) Insert at-least five records in table.
b) Create an Update Query to increase the cost of courses from 500 to 510. *Open the* **Create Update Query** *in Design view.*

## Procedure

### Create WorkshipList (workshopid, workshop, Cost)

1. Create a table in design view.
2. Use the same fields as above mentioned.
3. Save the table as *Student*.

Workshop List : Table

| Field Name | Data Type |
|---|---|
| WorkshopID | AutoNumber |
| Workshop | Text |
| Cost | Number |

### Insert records in table

4. Switch to Datasheet View of table & enter data in all fields for five records

Workshop List : Table

| WorkshopID | Workshop | Cost |
|---|---|---|
| 1 | Access | 500 |
| 2 | Excel | 500 |
| 3 | Powerpoint | 500 |
| 4 | Word | 500 |
| (AutoNumber) | | 0 |

Record: 4 of 4

### Create an Update Query to increase the cost of courses from 500 to 510.

1. Select **Create query in Design view** from Queries tab in Database window.
2. Add workshoptable table to the Query window.


3. Choose workshop, cost fields in the query.
4. Select the **Update To** row under the desired field and type the expression required to perform the update. *Type* **[Cost]+10** *into the* **Update To** *row.*

| | | |
|---|---|---|
| Field: | Workshop | Cost |
| Table: | Workshop List | Workshop List |
| Update To: | | [Cost]+10 |
| Criteria: | | |
| or: | | |

5. Run the query. ! A warning box appears stating that you cannot undo command.

6. Select **Yes**. The Microsoft Access warning box closes, and Access runs the update query and updates the records in the table.

## Practical 12

1. Create the table **Branch** with the following field.

Branch : Table

| Field Name | Data Type | Description |
|---|---|---|
| branchNo | Text | |
| Street | Text | |
| city | Text | |
| postcode | Text | |

2. Insert the following records

Branch : Table

| branchNo | Street | city | postcode |
|---|---|---|---|
| B002 | 22 Deer Rd | Lahore | SW1 4EH |
| B003 | 16 Amir st | Atak | AB2 3SU |
| B004 | 163 Main st | Gujranwala | G11 9QX |
| B005 | 32 Mansoor st | Bahawalpur | BS99 1NZ |
| B007 | 56 Dr. Road | Lahore | NW10 6EU |

Record: 1 of 5

3. Create table **Client** with the following field.

Client : Table

| Field Name | Data Type | Description |
|---|---|---|
| clientNo | Text | |
| fname | Text | |
| lname | Text | |
| telNo | Text | |
| prefType | Text | |
| maxRent | Text | |

4. Insert the following records in client table.

Client : Table

| clientNo | fname | lname | telNo | prefType | maxRent |
|---|---|---|---|---|---|
| CO56 | Aslam | Ali | 041-78777 | Flat | 800 |
| CR62 | Tahir | Mahmood | 0451-4443 | Flat | 1000 |
| CR74 | Tina | Muhammad | 042 87779 | House | 6000 |
| CR76 | Jamil | Karim | 041-6777 | Flat | 1000 |

Record: 1 of 4

5. Create a table **PrivteOwner** with the following field.

privateowner : Table

| Field Name | Data Type | Description |
|---|---|---|
| ownerNo | Text | |
| fname | Text | |
| lname | Text | |
| address | Text | |
| telNo | Text | |

6. Insert following **Records** in **Private Owner**.

privateowner : Table

| ownerNo | fname | lname | address | telNo |
|---|---|---|---|---|
| CO40 | Usman | khalil | 64 Well st,Gujranwala | 041-786555 |
| CO46 | Imran | Ali | 2 Fardous Atak AB2 7SX | 043-99898 |
| CO87 | salman | yousaf | 2 Park, Gujranwala G42 | 0431-454 |
| CO93 | Abdullah | Ejaz | 6 Asim st, Gujranwal G4 9QR | 0431-6456 |

Record: 2 of 4

7. Create a table **PropertyForRent** with the following field.

**propertyForRent : Table**

| Field Name | Data Type | Description |
|---|---|---|
| propertyNo | Text | |
| street | Text | |
| city | Text | |
| postcode | Text | |
| type | Text | |
| rooms | Number | |
| rent | Number | |
| owerNo | Text | |
| staffNo | Text | |
| branNo | Text | |

8. Insert the following records in **PropertyForRent** table.

**propertyForRent : Table**

| propertyNo | street | city | postcode | type | room | rent | owerNo | staffNo | branNo |
|---|---|---|---|---|---|---|---|---|---|
| PA14 | 161 Mian st | Atak | AB7 SSU | House | 6 | 6000 | CO46 | SA9 | B007 |
| PG16 | 51 Yasir st | Gujranwala | LAH-301 | Flat | 4 | 1000 | CO87 | SL41 | B005 |
| PG21 | 19 Mansoor st | Gujranwala | G11 9QX | Flat | 5 | 1000 | CO40 | | B003 |
| PG36 | 10 deer st | Gujranwala | G32 4QX | Flat | 3 | 800 | CO93 | SG37 | B003 |
| PG4 | 17 abdul st | Gujranwala | G12 9AX | House | 3 | 6000 | CO87 | SG37 | B003 |
| PL94 | 8 st col | Lahore | NW2 | Flat | 4 | 800 | CO93 | SG14 | B003 |

Record: 6 of 6

9. Create a table **Registration** with the folloiwng field.

**Registration : Table**

| Field Name | Data Type | Description |
|---|---|---|
| clientNo | Text | |
| branNo | Text | |
| staffNo | Text | |
| datejoined | Date/Time | |

10. Insert the following records in **Registration** table.

**Registration : Table**

| clientNo | branNo | staffNo | datejoined |
|---|---|---|---|
| CR56 | B003 | SG37 | 11/04/2000 |
| CR62 | B007 | SA9 | 07/03/2000 |
| CR74 | B003 | SG37 | 16/11/1999 |
| CR76 | B005 | SL41 | 02/01/2001 |

Record: 1 of 4

11. Create a table **Staff** with the following field.

**staff : Table**

| Field Name | Data Type | Description |
|---|---|---|
| staffNo | Text | |
| fname | Text | |
| lname | Text | |
| position | Text | |
| sex | Text | |
| DOB | Text | |
| Salary | Number | |
| branNo | Text | |

12. Insert the following Records in Staff table.

**staff : Table**

| staffNo | fname | lname | position | sex | DOB | Salary | branNo |
|---|---|---|---|---|---|---|---|
| SG37 | Abdullah | Ejaz | Assistant | M | 10-Nov-60 | 12000 | B003 |
| SG5 | Sultana | Babar | Manager | F | 3-Jun-40 | 24000 | B003 |
| SL21 | Nazia | khalil | Manager | F | 1-oct-45 | 30000 | B005 |
| SL212 | Hina | Ahmsd | Supervisor | F | 24-Mar-58 | 18000 | B003 |
| SL41 | Fatima | Luqman | Assistant | F | 13-Jun-65 | 9000 | B005 |

Record: 1 of 5

13. Create a table Viewing with the following field.

viewing : Table

| Field Name | Data Type | Description |
|---|---|---|
| clientNo | Text | |
| propertyNo | Text | |
| viewDate | Date/Time | |
| comment | Text | |

14. Insert the following Records

viewing : Table

| clientNo | propertyNo | viewDate | comment |
|---|---|---|---|
| CR56 | PA14 | 24/05/2001 | too small |
| CR56 | PG36 | 28/04/2001 | |
| CR56 | PG4 | 26/05/2001 | |
| CR62 | PA14 | 14/05/2001 | no dinning roo |
| CR76 | PG4 | 20/04/2001 | too remote |

Record: 1 of 5

## Relationship of the above Relations

**Branch** (branchno, street, city, postcode)

**Staff** (staffNo, fName, lname, Position, Sex, DOB, Salary, BranchNo)

**PropertyForRent** (PropertyNo, Street, city, postcode, type, rooms, rent, ownerNo, StaffNo, branchNo)

**Client** (ClientNo, fname, lname, telNo, PrefType, MaxRent)

**Private Owner** (ownerNo, fName, lname, address, TelNo)

**Viewing** (clientNo, propertyNo, Viewdate, Comment)

**Registration** (ClientNo, branchNo, staffNo, dateJoined)

## Perform the following Queries

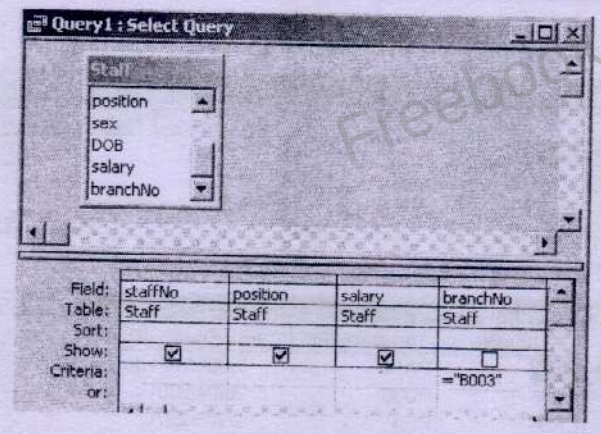
- Retrieve the branch number and address for all branch offices.

Query1 : Select Query

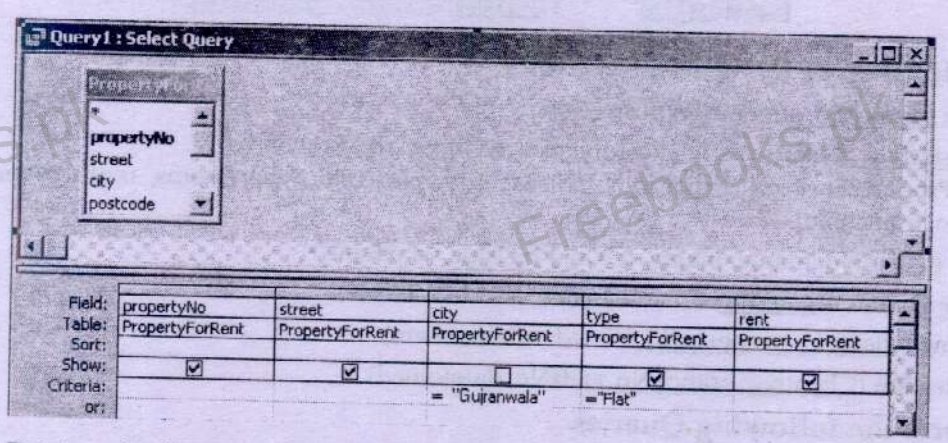
Branch
*
branchNo
street
city
postcode

| | | | | |
|---|---|---|---|---|
| Field: | branchNo | street | city | postcode |
| Table: | Branch | Branch | Branch | Branch |
| Sort: | | | | |
| Show: | ☑ | ☑ | ☑ | ☑ |
| Criteria: | | | | |
| or: | | | | |

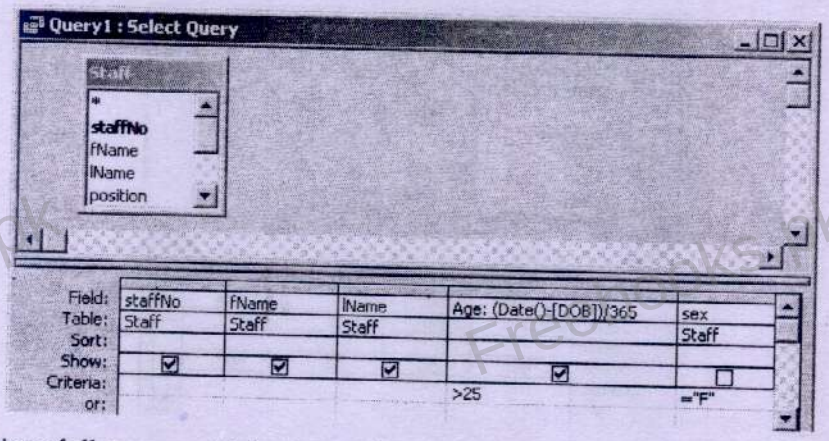
- Retrieve the staff number, position, and salary for all members of staff working at branch office B003.



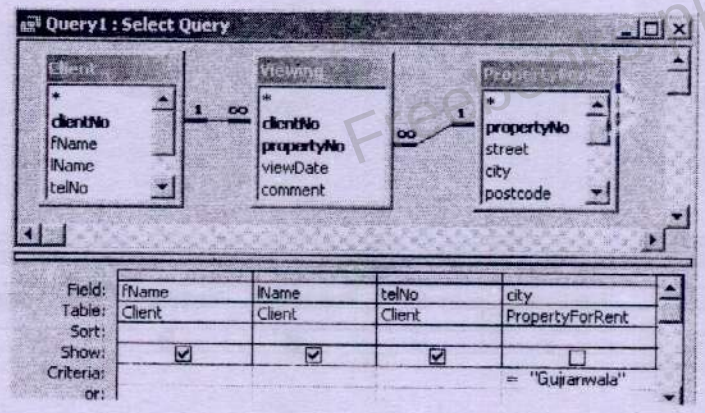

- Retrieve the details of all flats in Gujranwala.



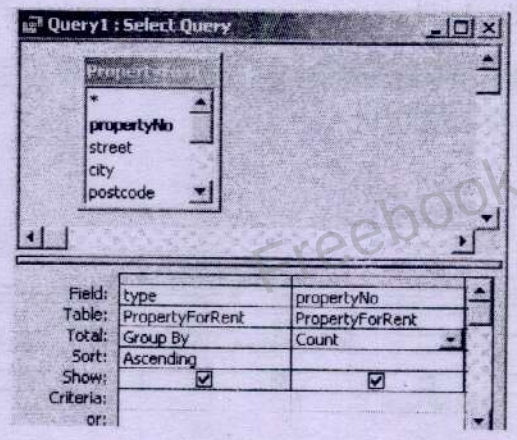

- Retrieve the details of all female members of staff who are older than 25 years old.



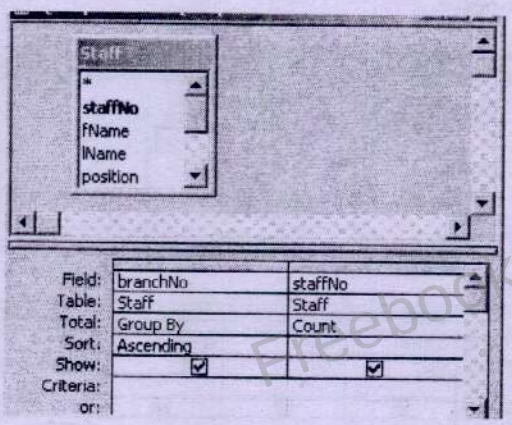

- Retrieve full name and telephone of all clients who have viewed flats in Gujranwala.




- Retrieve the total number of properties, according to property type.

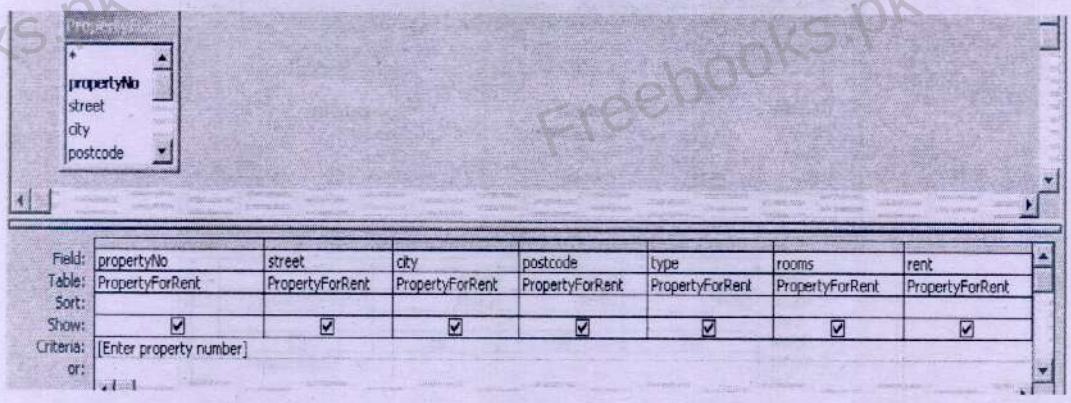



- Retrieve total number of staff working at each branch office ordered by branch number.

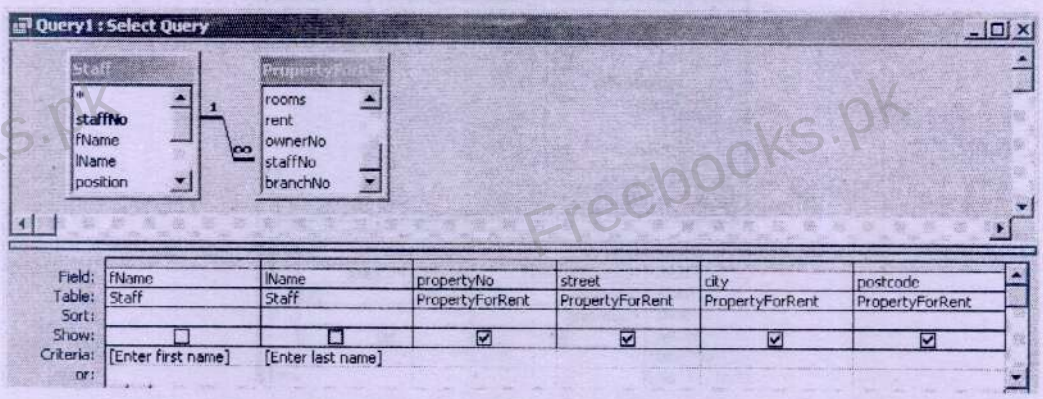



## Create the following additional advanced QBE queries for sample tables

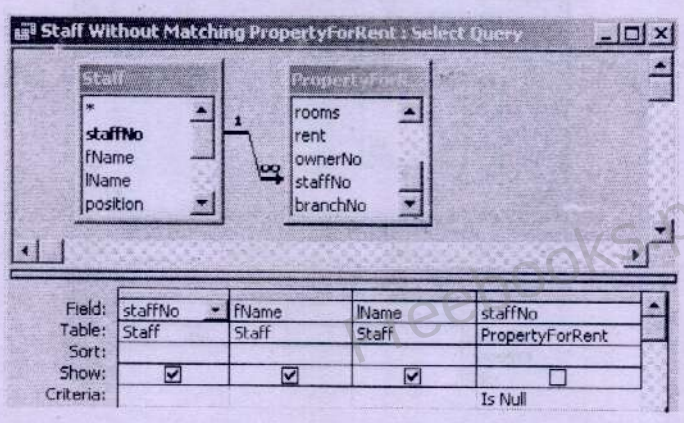
**(a)** Create a parameter query that prompts for a property number and then displays the details of that property.




(b) Create a parameter query that prompts for first and last names of a member of staff and displays the details of the property that the member of staff is responsible for.



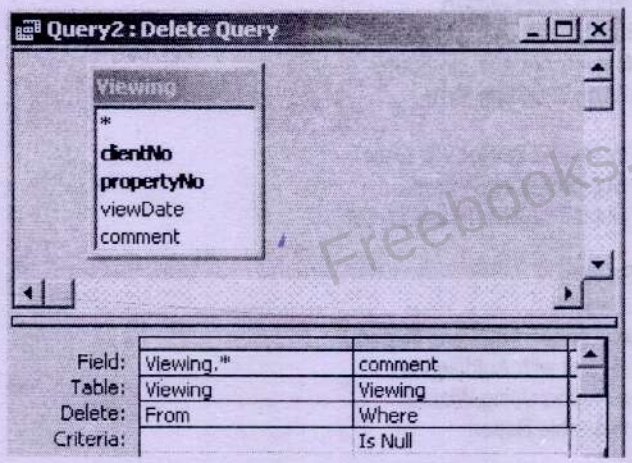

(c) Use **Find Unmatched** query to identify those members of staff who are not assigned to manage property.



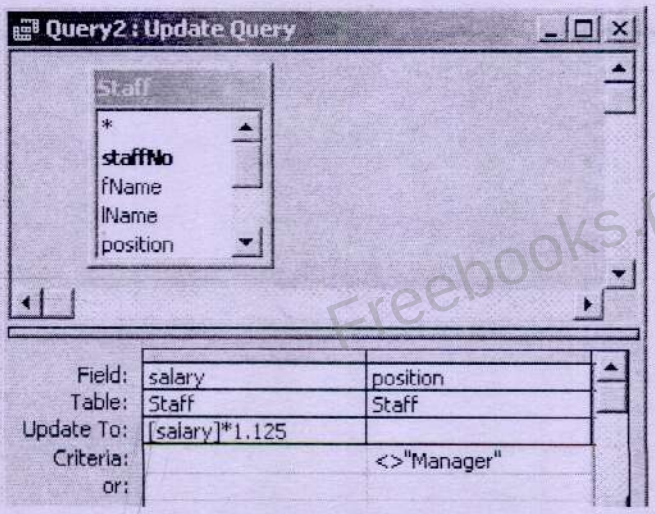

## Use action queries to carry out the following tasks on the sample tables

(a) Create a table **PropertyGujranwala** from **PropertyForRent** table with PropertyNo, street, postcode and type fields of the original table and contains only the details of properties in Gujranwala.


(b) Remove all records of property viewings that do not have an entry in Comment field.




(c) Update the salary of all members of staff, except Managers, by 12.5%.



CHAPTER 14

# Forms and Reports

## Chapter Overview

# 14.1 Form

A window that consists of visual components is called **form**. Forms are used to interact with databases through graphical user interface.

A form is constructed from a collection of individual design elements. These elements are called **controls** or **control objects** or **tools**. The common elements include buttons, check boxes, list boxes and radio buttons etc. Different elements are used for different purposes. A textbox is used to enter data and a label is used to display message to the user.

## Uses of Form

Form is used to manipulate databases easily. It can be used to:

- Add data in the database
- Modify data in the database
- Delete data from the database
- Retrieve and view data from the database
- Search the required data from the database

## 14.1.1 Types of Forms

MS Access provides the following types of forms:

### 1. Columnar Form

Columnar form is used to display one record at a time. It displays textboxes and labels. The textboxes represents the fields of the table or query. The labels represent names of fields. Columnar form provides different buttons to navigate through different records at bottom.

### 2. Tabular Form

Tabular form is used to display many records at one time. It displays records as a table. Each row in this form displays one record of the table. The labels are displayed on the top of each column. Tabular form provides different buttons to navigate through different records at the bottom.

### 3. Datasheet Form

Datasheet form is used to display many records at one time. It displays records in datasheet view of Access. Each row in this form displays one record of the table. The labels are displayed on the top of each column. Datasheet form provides different buttons to navigate through different records at the bottom.

### 4. Justified Form

Justified form is used to display one record at a time. The fields are justified according the form window. The labels are displayed on the top of each column. Justified form provides different buttons to navigate through different records at the bottom.

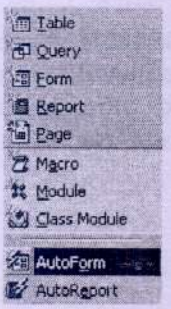
## 14.1.2 Creating Form using AutoForm

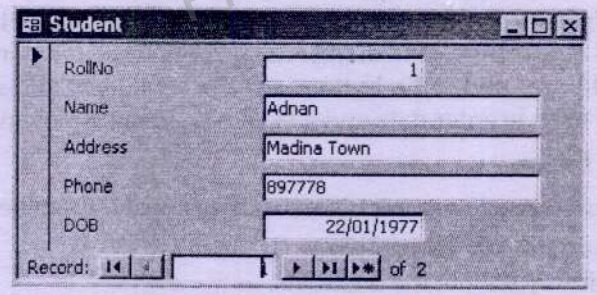
The simplest method of creating a form is to use **AutoForm**. To create an AutoForm, you simply select a data source, such as a table or query, and then select AutoForm. Access creates a simple form using all the fields in the data source laid out in a single column.

### Creating Form using AutoForm

- Select the table for which you want to create the form

- Click **AutoForm** on **Database** toolbar. If AutoForm button is not displayed, click the arrow and select it from the drop-down menu). OR click **Insert > AutoForm**. A form is created and opened in a new window.




## 14.1.3 Saving and Closing AutoForm

**AutoForm** is not saved as a database object automatically. If the form is closed without saving it, Access will prompt to save before closing.

- Click **Save** on **Database** toolbar **OR**
- Click **File > Save**. The **Save As** dialogue box is displayed.


- Change **Form Name:** if necessary and click **OK**. The form will be saved.
- Click **Close** icon on title bar to close the form.

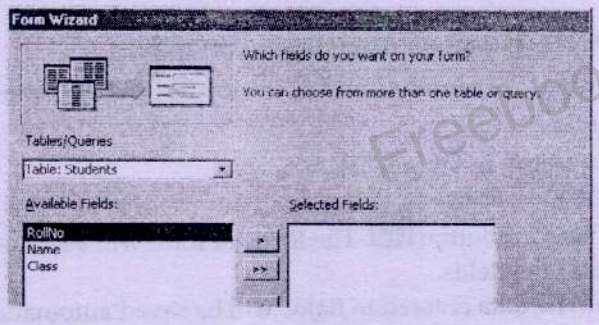
OR

- Click **File > Close**.
- Click **Forms** object Forms to open or edit the form again

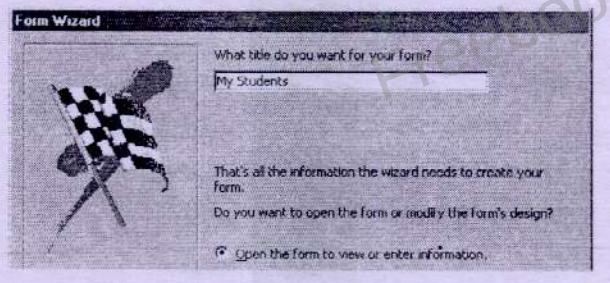

## 14.1.4 Creating Form using Wizard

The procedure to create form by using wizard is as follows:

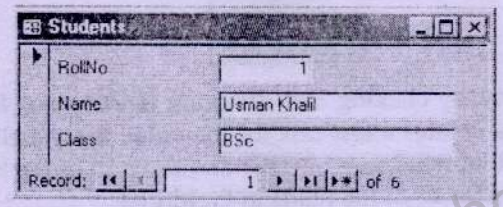
1. Open the database.
2. Click on **Forms** button in **Objects** list.
3. Double click **Create form by using wizard**. The wizard will appear:




4. Select any table from **Tables/Queries** list box. The fields of the selected table will appear in **Available Fields** box.
5. Click on any field to include in the form.
6. Click on > button. The field will move to **Selected Fields** box. OR click on >> button to include all fields in the form.
7. Click on **Next** button.
8. Select any layout for the form.
9. Click **Next** button.
10. Select any style for the form.
11. Click **Next** button. The following window will appear:




12. Enter any title.
13. Select **Open the form to view or enter information** radio button.
14. Click **Finish** button. The form will appear as follows:




## 14.1.5 Editing Records through Forms

The following procedure is used to edit records through forms:

- Open the database.
- Click on **Forms** button in **Object** list.
- Double click the form to open.
- Go to the record to be edited by using navigation buttons at the bottom.
- Change the contents of the fields.
- Close the form. The changes will be updated automatically.

### 14.1.6 Adding New Record through Forms

The following procedure is used to add new records through forms:

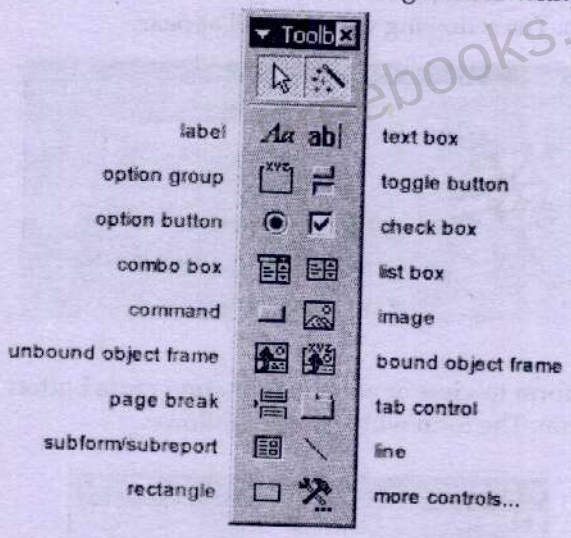
- Open the database.
- Click on **Forms** button in **Object** list.
- Double click the form to open.
- Click the **New Record** button. The fields of form will become empty.
- Enter new data in the fields.
- Close the form. The data entered in fields will be saved automatically.

### 14.1.7 Form Creation in Design View

Forms can be created in design view using different controls. This method allows the user to design the form according to his particular requirements. A toolbox is available in design view that provides different control such as textboxes and buttons etc.

The following procedure is used to create a form in design view:

1. Click **New** button on the form database window.
2. Select **Design View** and choose table or query to which the form will be associated.
3. Select **View > Toolbox** from to view the floating toolbar with additional options.

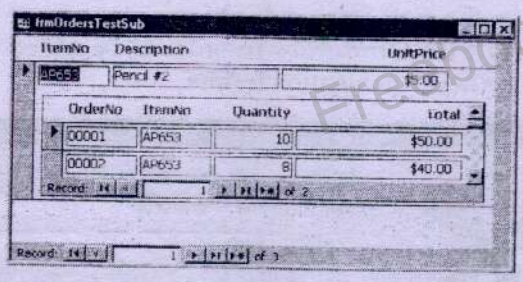



4. Add controls to form by clicking and dragging field names from **Field List** floating window. MS Access creates a text box for the value and label for the field name. OR Double-click **Field List** window's title bar and drag all of the highlighted fields to the form to add controls for all of the fields in **Field List**.

## 14.2 Subform

A **subform** is a form that is placed in a parent form. The parent form is also called **main form**. Subform is also known as **child form**. Subforms are particularly useful to display data from tables and queries with one-to-many relationships.

The following example displays data on the main form from an **item information** table. The subform contains all orders for that item. The item record is the **one** part of one-to-many relationship and orders are **many** side of the relationship. It means that many orders can be placed for the one item.




The subforms can be created in three ways:

- Creating a form and subform at once
- Creating subform using Subform wizard
- Creating subform using drag-and-drop method

## 14.2.1 Creating Form and Subform at Once

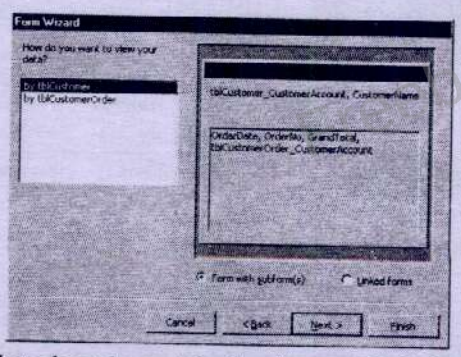
This method is used if no form has already been created. The main form and subform can be created automatically using the form wizard if:

- Table relationships are set properly OR
- A query involving multiple tables is selected.

For example, a relationship can be set between a table that contains customer information and a table that contains customer orders. The orders for each customer can be displayed together using a main form and subform.

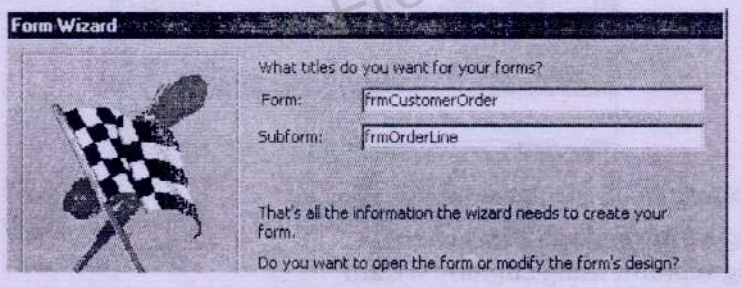
The following procedure is used to create a subform within a form:

1. Double-click **Create form by using wizard** on database window.
2. Select the first table or query from which the main form will display data from **Tables/Queries** menu.
3. Select fields that should appear on form by highlighting the field names in **Available Fields** list on left and clicking single arrow > button OR click double arrows >> to choose all of the fields.
4. Select another table or query from **Tables/Queries** drop-down menu and choose the fields that should appear on form.
5. Click **Next** after selecting all fields.
6. Select **Form with subform(s)** if the forms should appear on same page OR **Linked forms** if there are many controls on main form and a subform will not fit.
7. Click **Next.**




8. Select a tabular or datasheet layout for the form and click **Next**.
9. Select a style for the form and click **Next.**

10. Enter the names for the main form and subform.
11. Click **Finish** to create the forms. New records can be added in both tables or queries at once by using the new combination form.




## 14.2.2 Creating Subform using Wizard

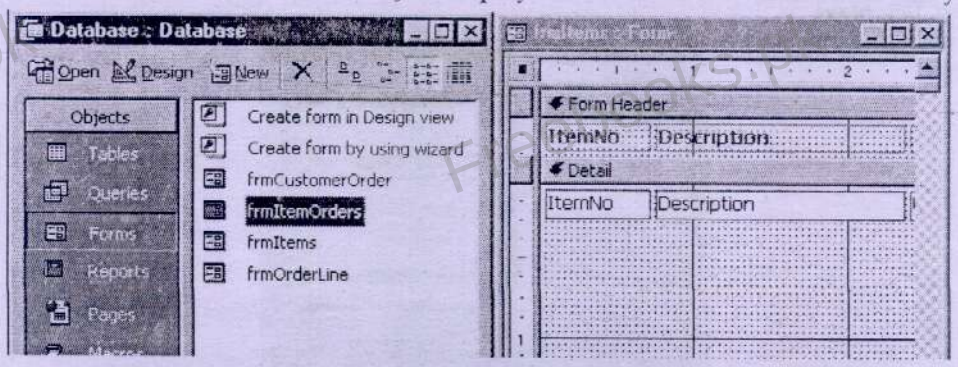
If main form or both forms already exist, subform wizard can be used to combine the forms. The following procedure is used to create a subform using subform wizard:

1. Open main form in Design View.
2. Make sure that **Control Wizard** button on the toolbox is pressed.
3. Click **Subform/Subreport** icon on toolbox and draw the outline of subform on main form. The Subform Wizard dialog box appears when mouse button is released.
4. Select **Use existing Tables and Queries** if the subform has not been created yet. OR select the existing form that will become the subform.
5. Click **Next**.
6. The next dialog window will display table relationships assumed by MS Access. Select one of these relationships or define different relationship.
7. Click **Next**.
8. Enter the name of subform and click **Finish**.

## 14.2.3 Creating Subform using drag-and-drop

This method is used to create subforms if two forms already exist. The relationships of tables must have already been set. The following procedure is used to create subform using drag-and-drop method:

1. Open main form in Design View.
2. Select **Window > Tile Vertically** to display database window and form side-by-side.




3. Drag form icon beside the name of subform on detail section of main form design.

## 14.3 Reports

**Reports** are the output of a database application. The user can generate different types of reports by manipulating the database. The information on the reports is arranged in different styles. The information displayed on the report can be from one or multiple tables or queries. The reports may contain graphs and charts etc.

The user cannot edit the data displayed on the reports. The user also cannot input data in reports. The reports are generated for printing purpose. They can also be displayed on screen and stored on the disk.

### 14.3.1 Uses of Reports

The reports are basically used for the following purposes:

- Reports present the required information in formatted style.
- Reports provide flexibility to present the same data in different ways.
- Reports can display information with graphics and charts etc.
- Reports are very important in making important decisions.
- Reports can be used to improve the database application.
- Reports can display the result of a query.

### 14.3.2 Types of Reports

The standard types of reports in MS Access are as follows:

#### 1. Columnar Reports

The **columnar reports** display the values of each field in each record of a table or query in one long column of text boxes. The labels indicate the name of the field. The text box to the right of label provides the values. The columnar report spreads the information for a single record over many rows.

#### 2. Tabular Report

The **tabular reports** provide a column for each field of the records in rows under column header. Additional pages are printed in sequence if columns do not fit on one page.

### 14.3.3 Difference between Form and Report

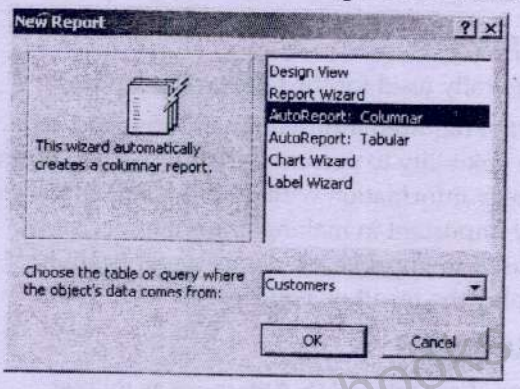
Form and report are important part of a database application. These are connected with one more tables. The main difference between a form and report is as follows:

| Form | Report |
|---|---|
| The basic purpose of form is to input data in tables. | The basic purpose of report is to display data from tables. |
| The data in form can be deleted. | The data in report cannot be deleted. |
| The data in form can be modified. | The data in form cannot be modified. |
| The forms are used on computer screen only. | The reports are normally used in printed form on the paper. |
| The data in form cannot be formatted. | The data in report can be formatted in different styles. |
| The user can add new data through form. | The user cannot add new data through report. |

## 14.3.4 Creating Report with AutoReport

**AutoReport** creates a columnar report based on specified table or query. The following procedure is used to create a report using AutoReport:

1. Open the desired database.
2. Click on **Report** button from **Objects** list.
3. Click on **New** button. **New Reports** dialog box will appear.
4. Select columnar or tabular report option.
5. Click on the list box at the bottom of the dialog box. A list will appear.



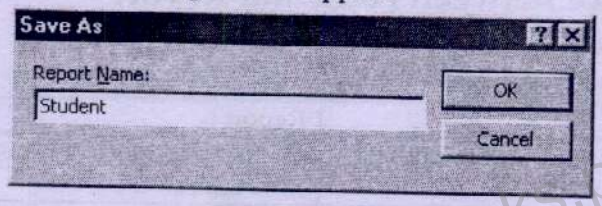

6. Select the desired table or query.
7. Click OK button. The report will appear.

The format of the report output is very basic. MS Access does not provide any formatting of the page header or footer. The user can modify the report in design view.

### Saving and Closing an AutoReport

The **AutoReport** is not saved to the database automatically. The following procedure is used to save the report:

1. Select **File > Close.** You are prompted to save the report.
2. Click **Yes.** The **Save As** dialog box will appear.

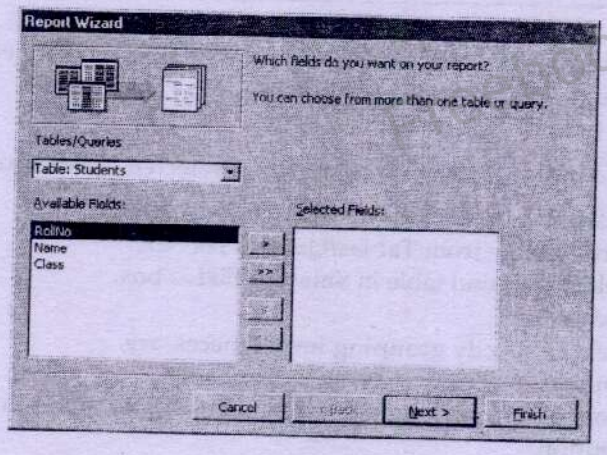



3. Enter a name and click **OK.** The report is saved to the **Database Window.**

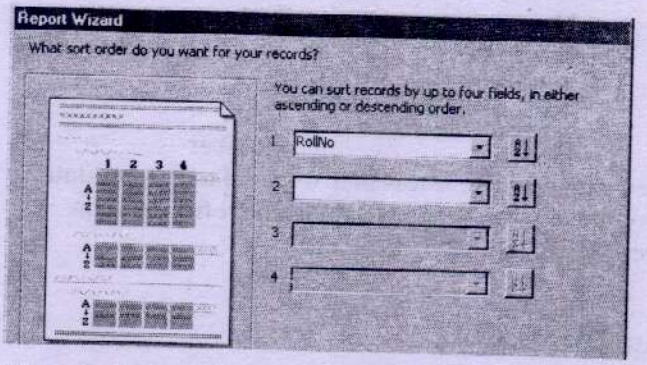
## 14.3.5 Creating a Report Using Wizard

The following procedure is used to create a single-table report using report wizard:

1. Open a database.
2. Click on **Reports** button in **Object** list.
3. Double click **Create report by using wizard.** The report wizard will appear.




4. Select a table from **Tables/Queries** list box. The fields of the selected table or query will appear in **Available Fields** box.
5. Click on any field to include in the report.
6. Click on [ > ] button. The field will move to **Selected Fields** box. OR
7. Click on [ >> ] button to include all fields in the report.
8. Click on **Next** button.
9. Select any field to specify **grouping level** if necessary.
10. Click **Next** button. The next window will appear.




11. Select any field according to which data record will be sorted in report.
12. Click **Next** button.
13. Select **layout** option and **orientation** option.
14. Click **Next** button.
15. Select any style for the report.
16. Type the name of the report.
17. Click **Finish** button. The report will appear.

## 14.3.6 Creating two-table Report

The following procedure is used to create a two-table report using report wizard:

1. Open a database.
2. Click on **Reports** button in **Object** list.
3. Double click **Create report by using wizard**. The report wizard will appear.

4. Select a table from **Tables/Queries** list box. The fields of the selected table or query will appear in **Available Fields** box.
5. Click on any field to include in the report.
6. Click on [>] button. The field will move to **Selected Fields** box. OR
7. Click on [>>] button to include all fields in the report.
8. Select the second table from **Tables/Queries** list box.
9. Move the fields of second table in **Selected Fields** box.
10. Click on **Next** button.
11. Select any field to specify **grouping level** if necessary.
12. Click **Next** button. The next window will appear.
13. Select any field according to which data record will be sorted in report.
14. Click **Next** button.
15. Select **layout** option and **orientation** option.
16. Click **Next** button.
17. Select any style for the report.
18. Type the name of the report.
19. Click **Finish** button. The report will appear on the screen.

## Practical 1

a. Create a table called Student:

| Field Name | Data Type |
|---|---|
| RollNo | Number |
| Name | Text |
| Class | Text |
| AdmDate | Date |
| Address | Text |
| Phone | Number |

b. Make sure that **AdmDate** is less than or equal to current date.
c. Create a columnar form to input data to insert five records.
d. Create a report using the Report wizard.

## Procedure

### a. Creating a Student Table

1. Create a table in design view.
2. Select **RollNo** as primary key.

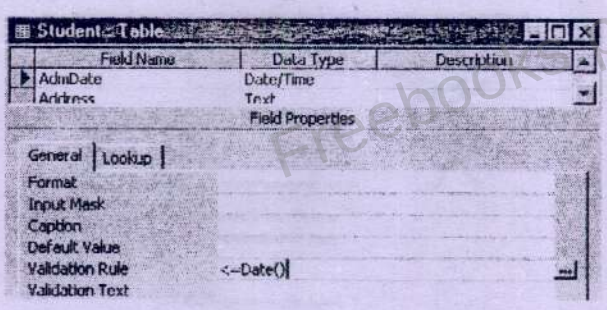
Student : Table

| Field Name | Data Type | Description |
|---|---|---|
| RollNo | Number | |
| Name | Text | |
| Class | Text | |
| AdmDate | Date/Time | |
| Address | Text | |
| Phone | Number | |

3. Save the table as **Student**.

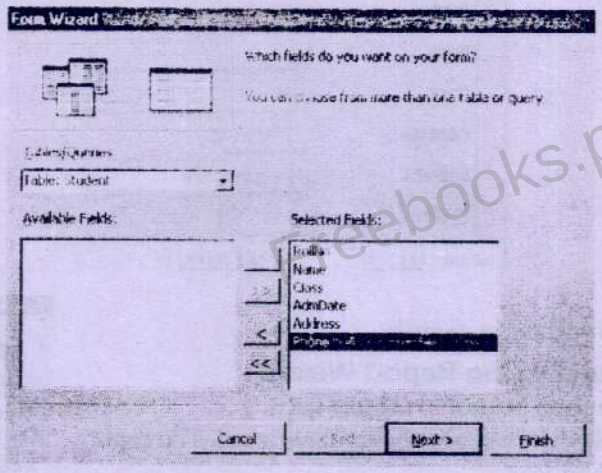
### b. Apply Validation Rule

1. Type <=*Date()* in **Validation Rule** property box of **Admdate** field.

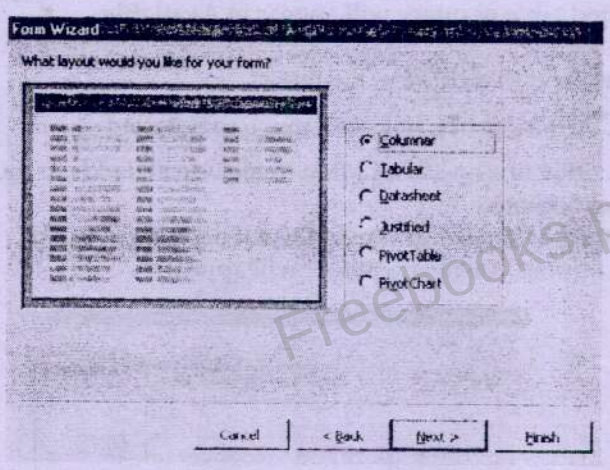



## c. Creating Columnar Form to Insert Five Records

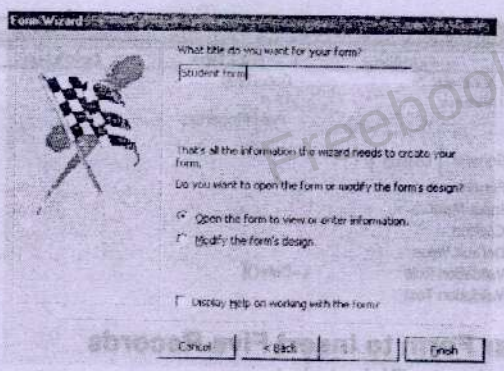
1. Select Forms option in **Objects** list.
2. Double click **Create form by using wizard** from in **Database** window. The **Form Wizard** will appear.
3. Select all the fields from **Student** table in **Selected fields**.



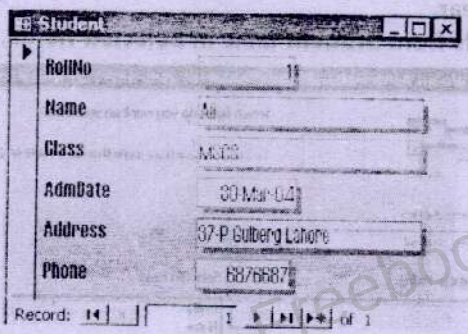

4. Select **Columnar** from layout to insert records.




5. Choose **International** style and at end type **Student form** as name of form.

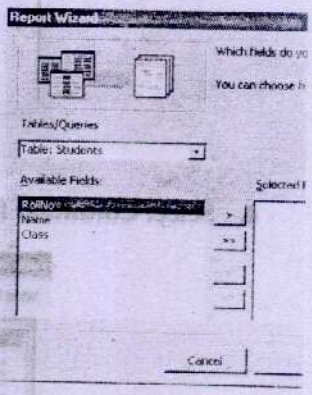



6. Click **Finish**. A form of required fields is created.
7. Fill all the fields as follows.

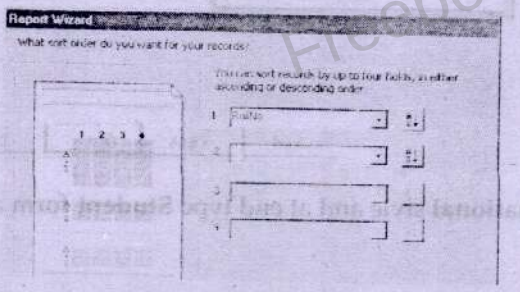



8. Click ▶* to add more records.

## d. Creating a Report using Report Wizard




1. Click on **Reports** button in **Object** list.
2. Double click **Create report by using wizard**. The report wizard will appear.
3. Select a student from **Tables/Queries** list box. The fields of the selected table or query will appear in **Available Fields** box.
4. Click on any field to include in the report.
5. Click on > button. The field will move to **Selected Fields** box. **OR** Click on >> button to include all fields in the report.
6. Click on **Next** button.
7. Select any field to specify grouping level if necessary.
8. Click **Next** button. The next window will appear.



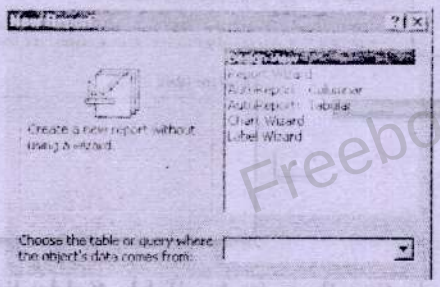
9. Select any field according to which data record will be sorted in report.
10. Click **Next** button.
11. Select layout option and orientation option.
12. Click **Next** button.
13. Select any style for the report.
14. Type the name of the report.
15. Click Finish button. The report will appear.

## 14.4 Label Report

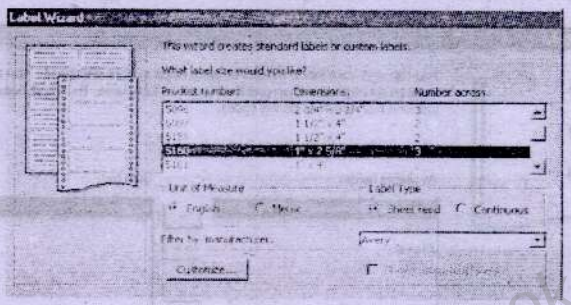
The **Label Wizard** provides a useful method to create reports to print information on labels.

### Create a Label Report

- Select **Reports** object from **Database Window.**
- Click **New** on **Database Window** toolbar OR
- Click **Insert > Report**. The **New Report** dialog box is displayed.

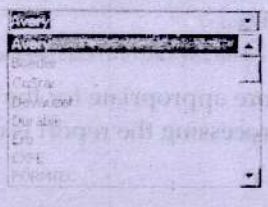



- Select **Label Wizard**.
- Choose the table or query from which the labels will be produced.
- Click **OK.** The **Label Wizard** is started.

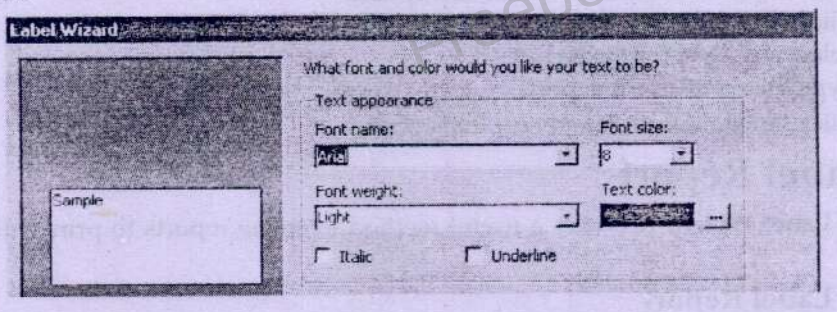



MS Access provides commonly available formats for lables. The user can select the manufacturer whose labels are to be used from **Filter by manufacturer:** list.

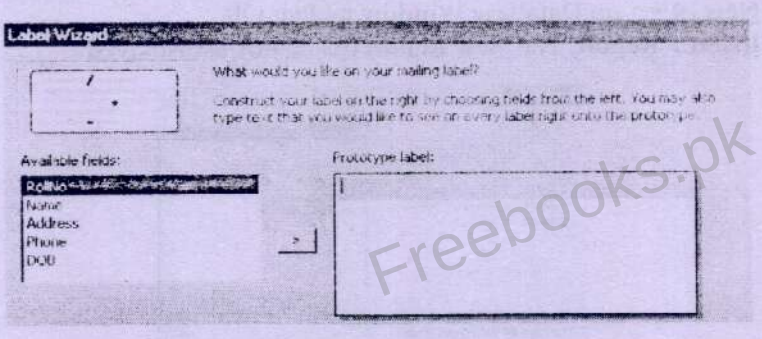
- Make the selection from **Filter by manufacturer** list



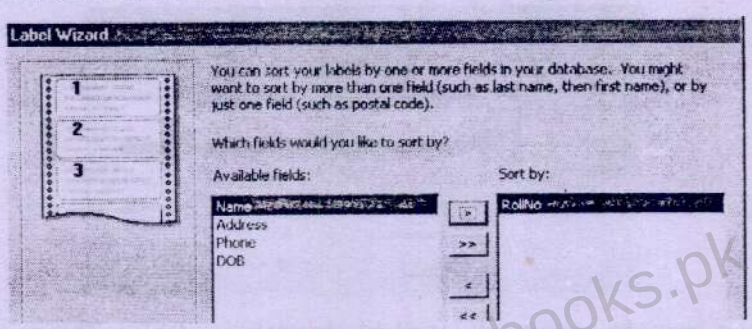
- **Unit of Measure** is used to select a label size by inches or centimeter units. **Label Type** is used to choose continuous or sheet feed labels. Choose the settings and click **Next >**



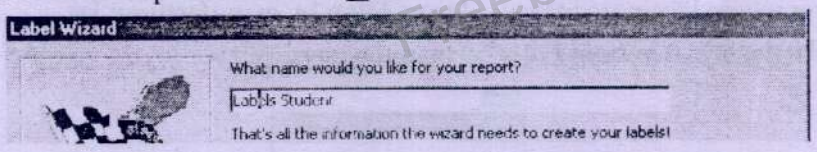

- Change the settings as appropriate and click **Next >**




- Position fields by selecting them in **Available fields:** list and clicking > to move them on label. The selected field is added to the label as {*FieldName*}. The user can add additional text around this field.
- Add additional fields and text as required and click **Next >**.




- Choose the sort options and click **Next >**




- Re-name the title to one more appropriate for the label report
- Click **Finish.** After some processing the report is displayed in **Print Preview**.
- Click **File > Close**.

## Practical 2

Create a table called Student having following fields:

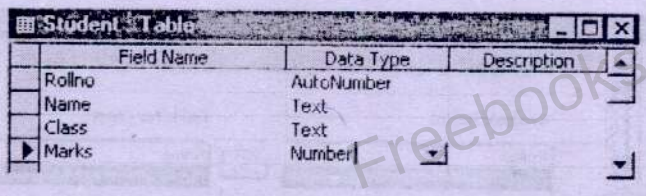
| Field Name | Type |
|---|---|
| Rollno | Autonumber |
| Name | Text |
| Class | Text |
| Marks | Number |

### Perform the following tasks:

a. Insert five records in the table.
b. Create **Column Chart** using report wizard to display student name and marks.
c. Set **Chart Title** as *'Student Result chart'*.

### Create 'Student' table (Rollno, Name, Class, Marks)

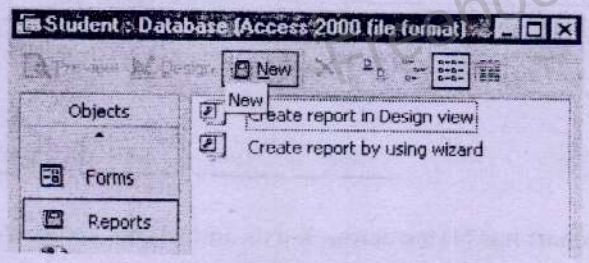
1. Create a table in design view.



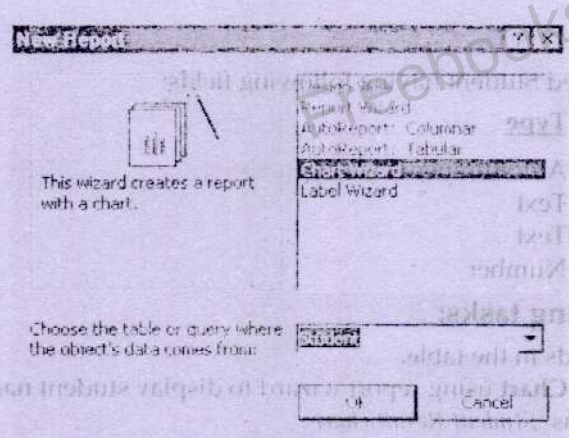

2. Save the table as **Student**.
3. Switch to **Datasheet View** of table and enter data for five records.


### Using Report wizard, create Columnar Chart to display student name and respective marks

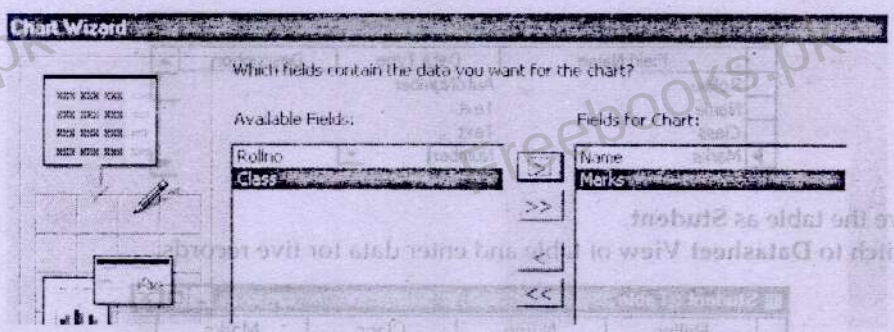
1. Select **New** from **Reports** tab in **Database** window. **New Report** dialog box appears.



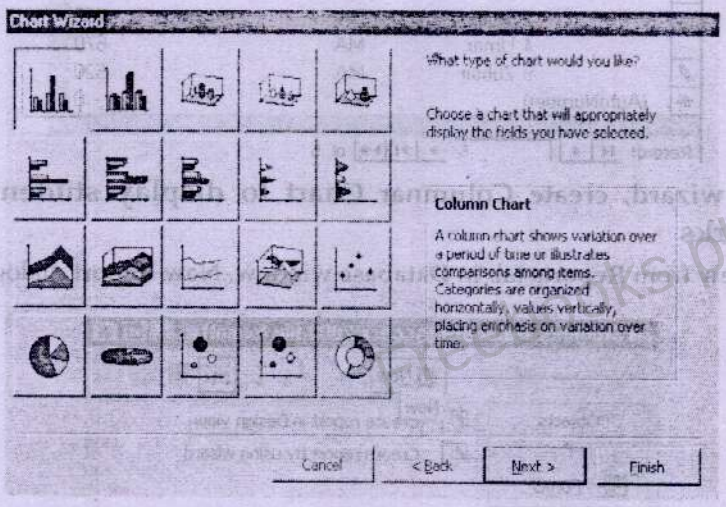

2. Select **Chart Wizard** from the given list and choose **Student** table from **Table or query** where the object's data comes from drop down list.



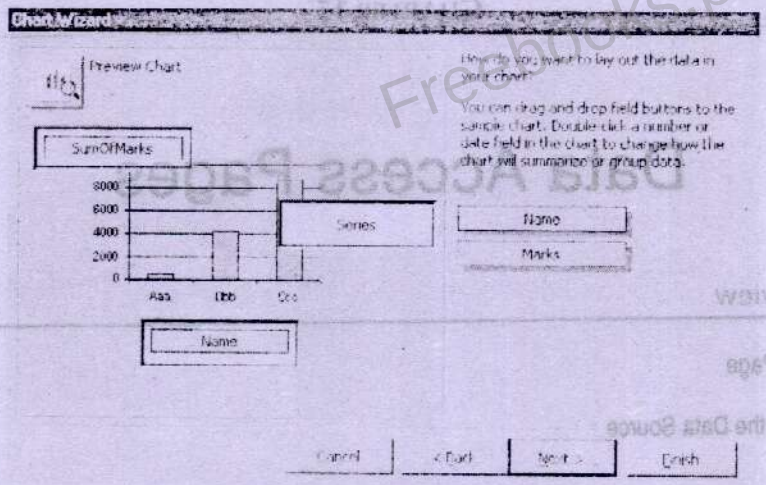

3. Click **OK** button to open **Chart Wizard**.
4. Select **Name** and **Marks** fields for **Chart** and press **Next >**.



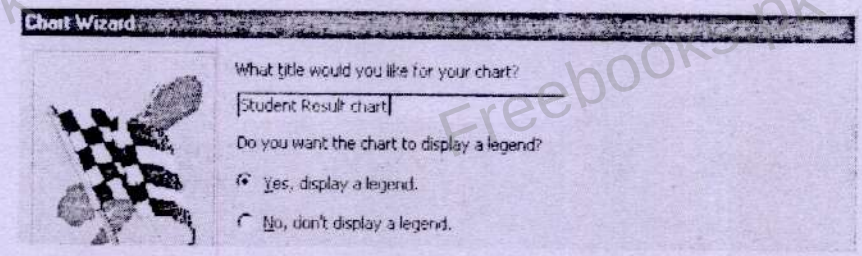

5. Choose **Column Chart** from the available types and press **Next >**.



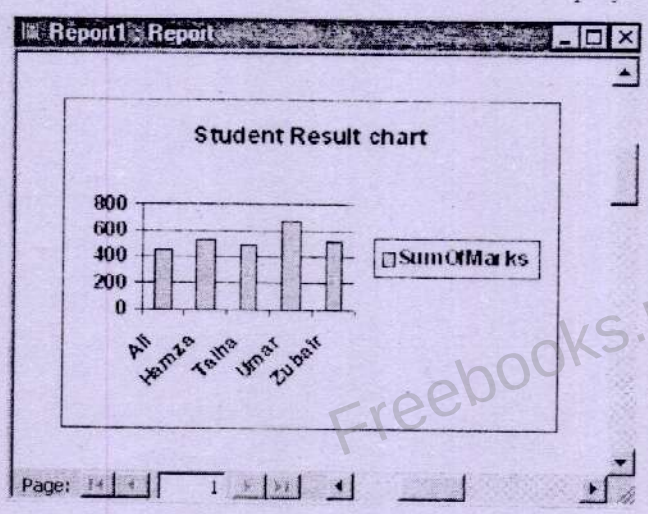

6. The layout for chart has **Name** across X-axis and **Marks** across Y-axis. Press **Next >**.




7. Set **Chart Title** as **Student Result chart** and press **Finish**.




8. A report chart showing Names and Marks of students is displayed in print preview.



CHAPTER 15

# Data Access Pages

## Chapter Overview

# 15.1 Data Access Page

Data Access Pages are web pages bound directly to the data in the database. They can be viewed in the web browser Microsoft Internet Explorer 5.

Note To get access to the full range of design features when creating a data access page, you will need Internet Explorer 5.5 or later installed.

Using Data Access Pages, you can:

Create HTML forms in Access and publish them to a web server for access over an intranet or internet. Any data entered or edited using these pages goes directly into database.

Create an interactive data report for users to view and analyze data through IE5. You can create pages with sorting and grouping levels (similar to reports) and add calculated fields. You can also include charts and PivotTable lists. Users can filter or sort data as well.

Unlike other database objects, a data access page is not stored in the database .MDB file. It is a separate HTML page, stored on the HTTP web server. The Access .MDB file does contain a shortcut to the page.

**Practical**

Create a new database 'Sale record' and create a table *'Client'* with following fields:

| Fieldname | Data type |
|---|---|
| Salesman No | Number |
| Name | Text |
| Sale | Number |
| City | Text |

## Perform the following tasks:

a. Add five records to above table.
b. Create a Data Access Page, using Page Wizard, containing all the fields of above table except *Salesman No.*
c. Set the page title as 'Workers'.
d. Apply 'Sumi Painting' theme to page.
e. Save the page as 'Workers Info'.
f. Switch to **Page View** and change *Name* of 2nd record.

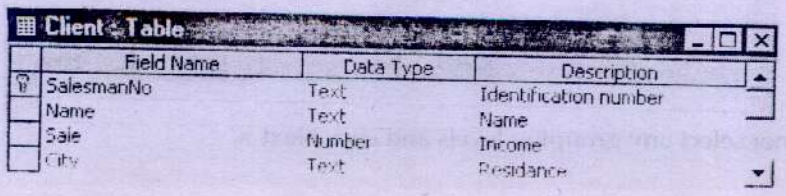
**Procedure**

## a. Create a new database named 'Sale Record'

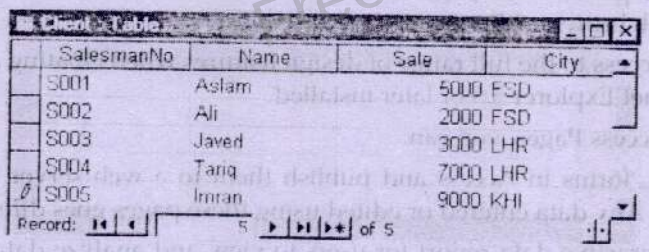
1. Open **MS Access.**
2. From **File** menu select **New** option & create a new database named *Sale Record.*

## b. Create 'Client' table (SalesmanNo, Name, Sale, City)

1. Create a table in design view.
2. Select SalesmanNo as **Primary Key.**



3. Save the table as Client.
4. Switch to Datasheet View of table & enter data as mentioned.

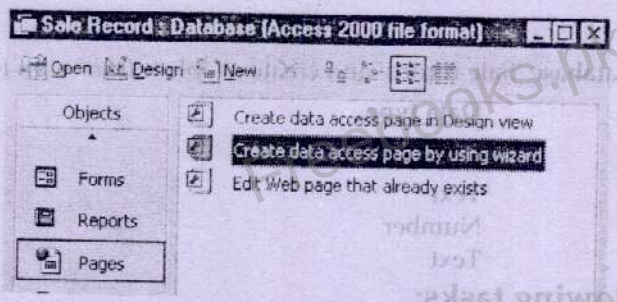


| SalesmanNo | Name | Sale | City |
|---|---|---|---|
| S001 | Aslam | 5000 | FSD |
| S002 | Ali | 2000 | FSD |
| S003 | Javed | 3000 | LHR |
| S004 | Tariq | 7000 | LHR |
| S005 | Imran | 9000 | KHI |

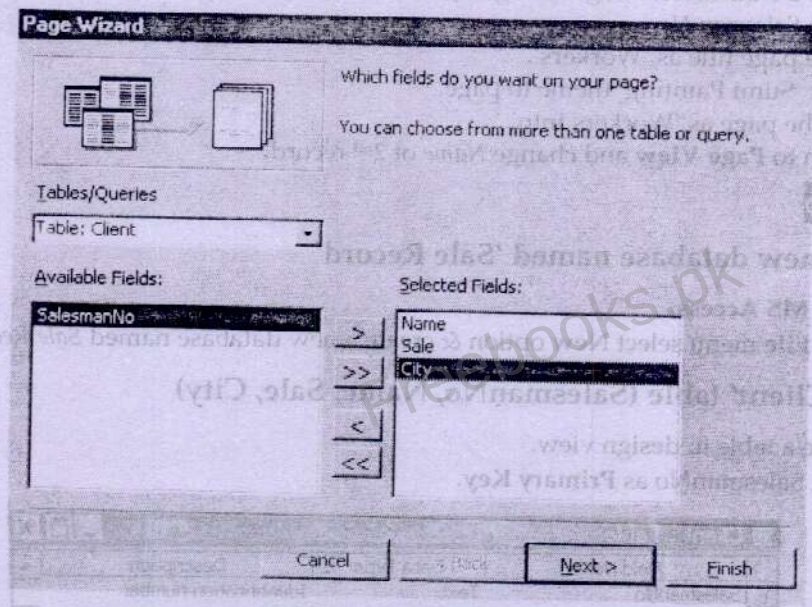
Record: 5 of 5

**c. Create a Data Access Page, using Page Wizard, containing all the fields of above table except *Salesman No***

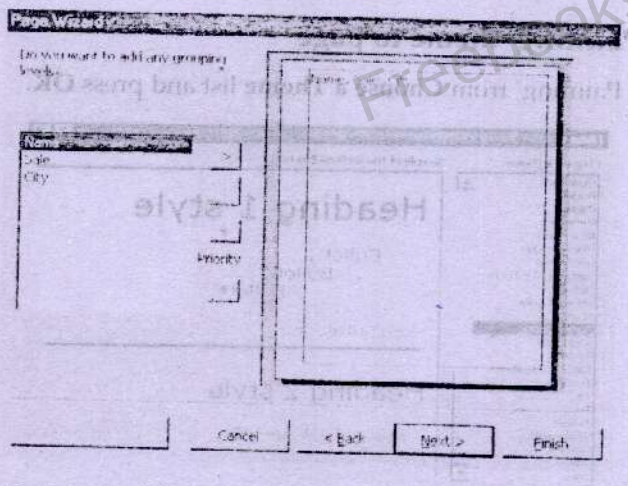
1. Select **Pages** object from **Database** window.
2. Double click **Create data access page by using wizard**. Page Wizard will open.



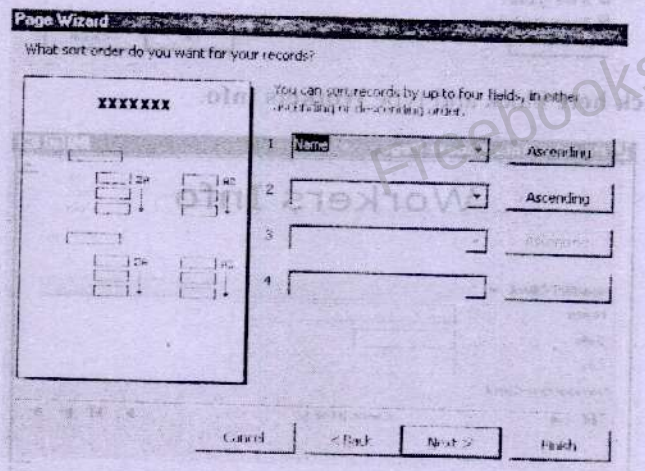

3. Select the fields **Name**, **Sale** and **City** from the table and click **Next >**.



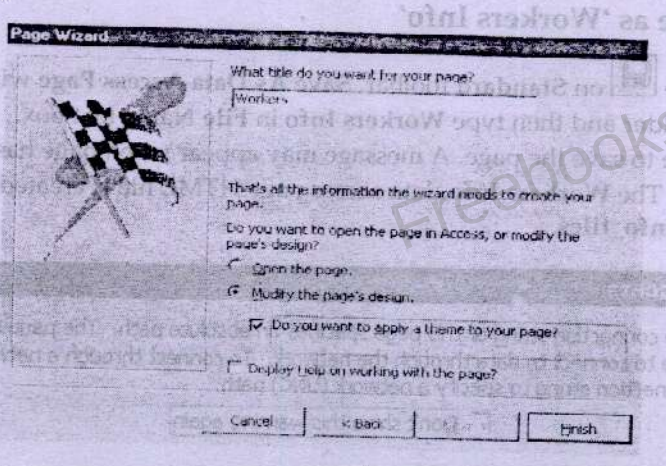

4. Do not select any grouping levels and click **Next >**.




5. Select **Name** field to sort the records in **Ascending** order and click **Next >**.



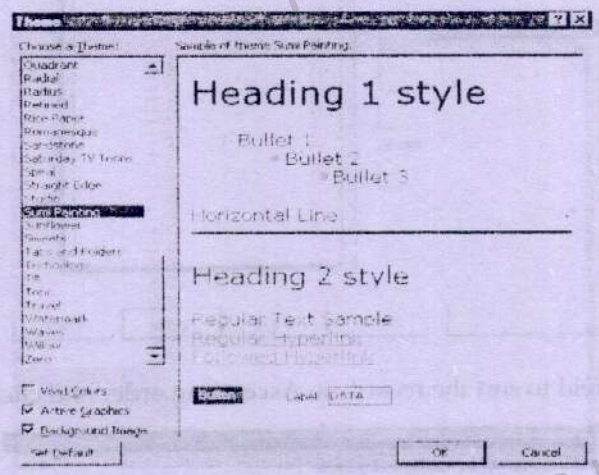

6. Type **Workers** as title of page and then check in the **Do you want to apply a theme to your page?** box.



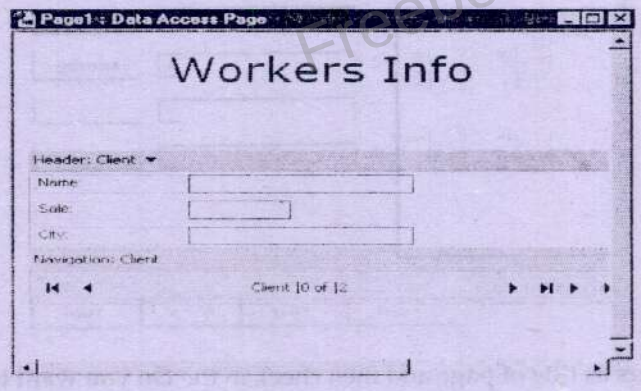

7. Press **Finish**.

## d. Apply 'Sumi Painting' theme to page

1. Select 'Sumi Painting' from **Choose a Theme** list and press **OK**.

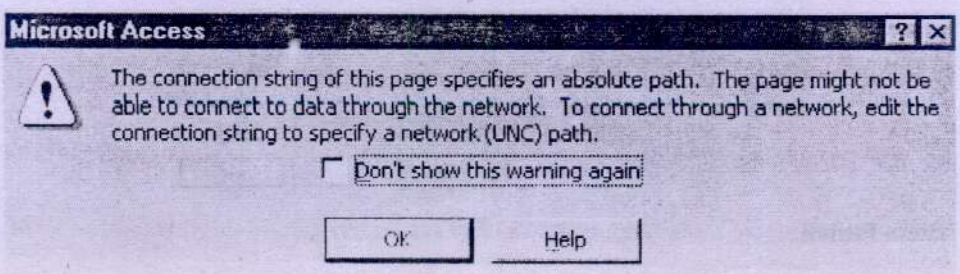



2. Click in **Click here** block and type **Workers Info**.



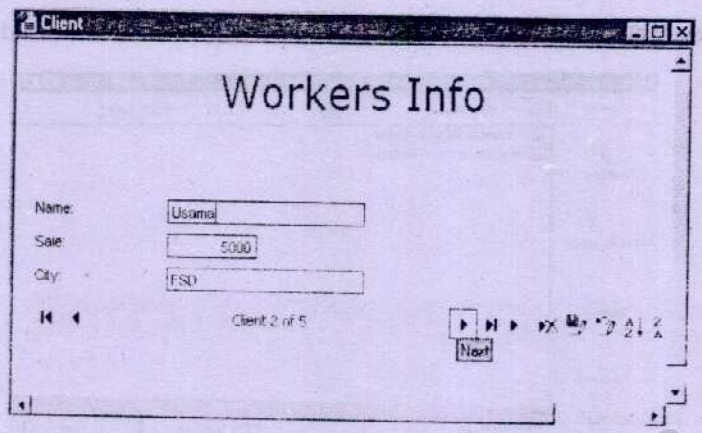

## e. Save the page as 'Workers Info'

1. Click **Save** on **Standard** toolbar. **Save As Data Access Page** window will appear.
2. Select a folder and then type **Workers Info** in **File Name** text box.
3. Press **Save** to save the page. A message may appear when page has been saved.
4. Press **OK**. The **Workers Info** data access page HTML file is created with a subfolder **Workers Info_files**.



### f. Switch to Page View and change *Name* of 2nd record

1. Click **View > Page View** from menu bar. Page will be opened in form view.
2. Move to second record by clicking **Next** button from toolbar at the end of page.
3. Click in **Name** field and change the name. This change is temporary and can't update database.




## 15.2 Connecting to the Data Source

A data access page can derive its data from a normal Access database (or from an SQL database). In order for users to access the data source, it must be stored on a shared drive with appropriate access permissions.

The ConnectionString property of the data access page stores information about where the data source is located. Alternatively, an easier way to maintain this information is by using a ConnectionFile, which can be re-used in other pages.

Note: Allowing access through a data access page creates a connection to the database (and potentially to your computer network), meaning that both the connection (the data access page) and the data source must be secured against unauthorized or malicious use. Do not allow use of data access pages over the internet without securing the connection properly.

**Practical 2**

Create a Connection File named Client Connection, for the previously created page, Workers Info.htm, connecting to Client database.

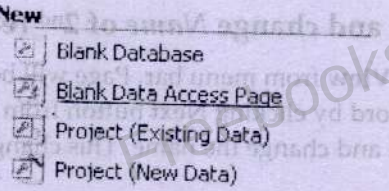
### Perform the following tasks:

a. Modify the **Connection File** property of *Workers Info* data access page.
b. Change *Name & City* fields of first record in Client table of *Sale Record* database.
c. Open the *Workers Info* page and see the effect of changes produced in database.

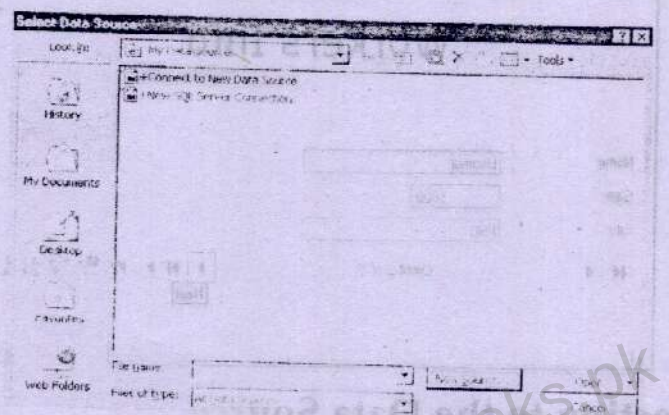
**Procedure**

### a. Creating a Connection File named *Client Connection*

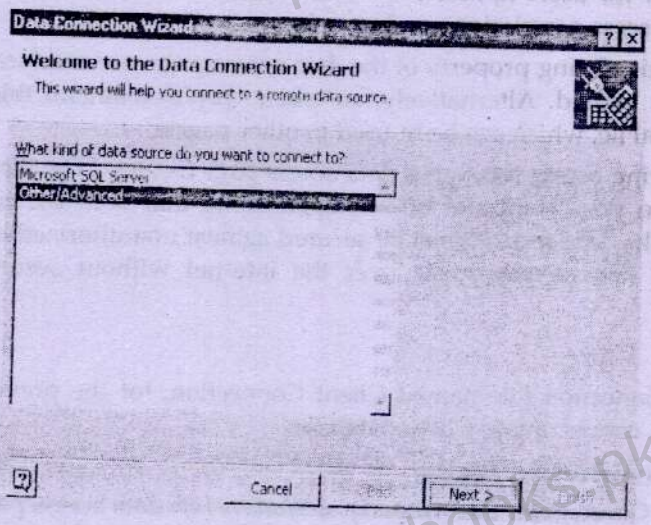
1. Open MS Access and click **File > New.**
2. Select **Blank Data Access Page** from **New File** task pane. The **Select Data Source** dialog box will open.



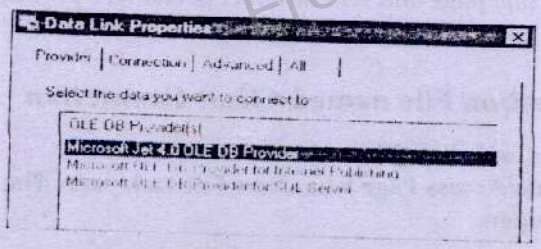

3. Click **New Source...** button. The **Data Connection Wizard** is started.



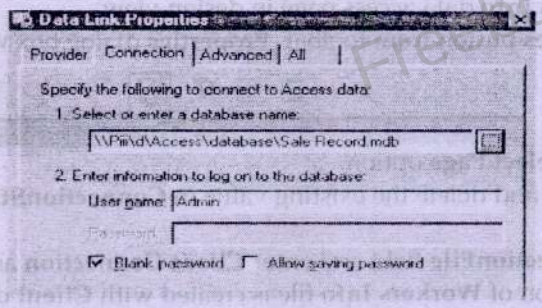

4. Select **Other/Advanced** and click **Next >**. The **Data Link Properties** dialog box is displayed.



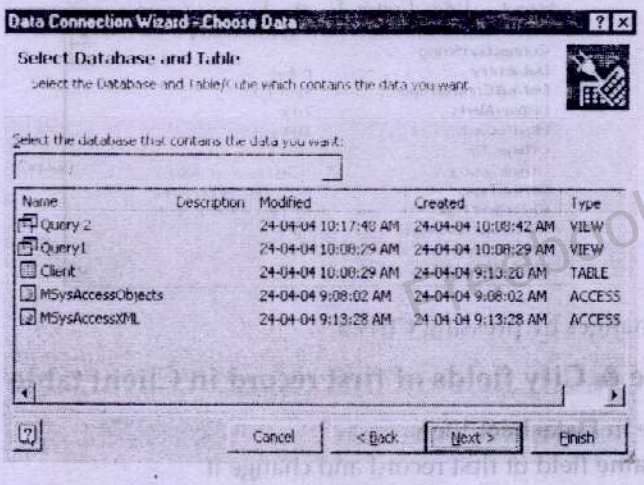

5. Select **Microsoft Jet 4.0 OLE DB Provider** and click **Next >>**.



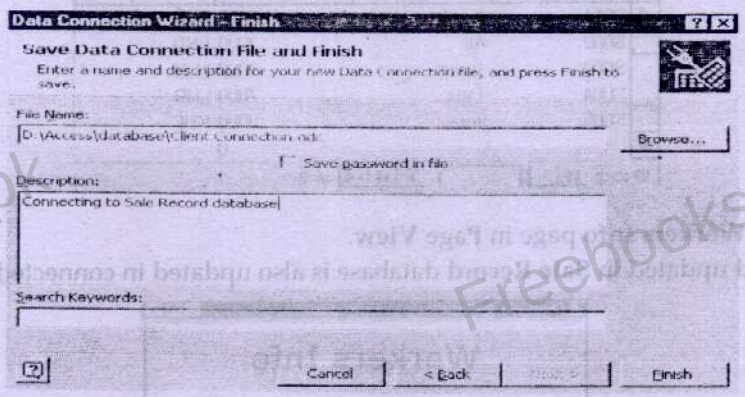
6. Enter the location of **Sale Record** database in **Select or enter a database name:** text box and press **OK**.




7. Press **Next >**.




8. Click **Browse** to save **Connection File** in a suitable location with the name **Client Connection**.
9. Enter a **Description** to indicate which database the file connects to.

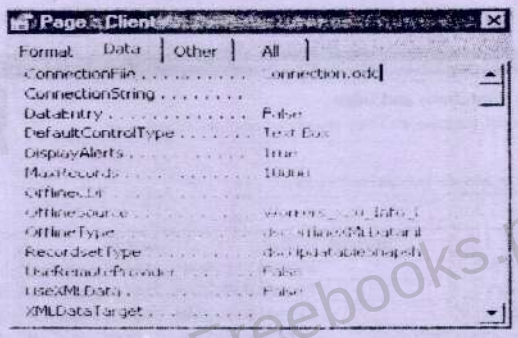



10. Click **Finish** button. The **Select Data Source** dialog box will appear.
11. Press **Cancel** button. **Blank data access page** will appear on screen.
12. Delete the blank data access page.

## b. Modify the Connection File property of Workers Info data access page

1. Open **Workers Info** data access page in design view.
2. Click **Properties** button from toolbar. **Properties** dialog box will appear.

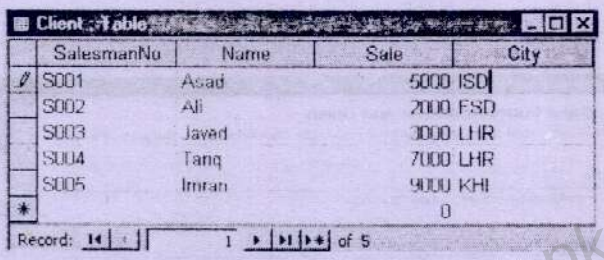

3. Click **Edit > Select Page** option.
4. Click **Data** tab and delete the existing value in **ConnectionString** field in **Properties** dialog box.
5. Click in **ConnectionFile** field and enter **Client Connection** as name of **Connection File**. Connection of **Workers Info** file is created with **Client** database.



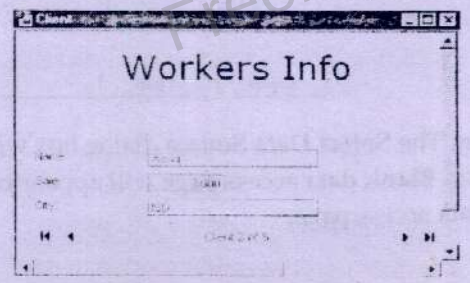

6. Save the changes by pressing **Ctrl+S**.

## c. Change Name & City fields of first record in Client table

1. Open table in **Datasheet** View.
2. Click in **Name** field of first record and change it.
3. Click in **City** field and change it.




4. Open **Workers Info** page in **Page View**.
5. Record updated in **Sale Record** database is also updated in connected page.



# MS Access Exercise

## Exercise # 1

**Create a table called student with the following Fields:**

| Fieldname | Data type |
|---|---|
| Rollno | Number |
| Stdname | Text |
| FatherName | Text |
| Birthdate | Date/Time |

a) Set Rollno as primary key.

b) Enter five records into the Student Table

c) In the student table ensure that that the roll no should never be below 0(zero). Also provide an appropriate message that will be display in case the user enters an invalid value.

## Exercise # 2

**Create a table called product:**

| Fieldname | Data type |
|---|---|
| Productno | Text |
| Description | Text |
| Unit_measure | Text |
| Cost_price | Number |

The data in table as follows:

| Productno | Description | Unit_measure | Cost_price |
|---|---|---|---|
| P001 | Monitor | Piece | 12000 |
| P002 | Mouse | Piece | 120 |
| P003 | CD Drive | Piece | 2000 |
| P004 | Keyboards | Piece | 200 |
| P005 | Printer | Piece | 5000 |

a) Create a query to display the product in the sort order of their description

b) Create a query to Find the products whose cost price are less than 1500

c) Create a query to display all the products which begin with the letter M and have cost price greater than 140.

## Exercise # 3

**Create two tables called Student and Result with the following fieldnames and data types:**

**Student**

| Field Name | Data Type |
|---|---|
| Rollno | Number |
| Name | Text |
| Address | Text |

**Result**

| Field Name | Data Type |
|---|---|
| Rollno | Number |
| Subject | Text |
| Marks | Number |

a) Use relational ships window to establish a relationship between the student and result tables where student is the parent table. The relationship should have referential integrity enforced, with the cascade update and delete settings enabled.

b) Enter Five Records in both tables.

## Exercise # 4

**Create a table called student with the following structure:**

| Field Name | Datatype |
|---|---|
| Rollno | Number |
| Stdname | Text |
| FatherName | Text |
| Date_of_birth | Date/Time |

a) Set Rollno as primary key.

b) Enter five records into the Student Table

c) Uses the form wizard to create form with a tabular format. Show the code behind the form. Modify the form to include an inch or two of blank space between the last field and the form footer section. Save the form as studentform

## Exercise # 5

**Create the following two tables**

**Employee**

| Field Name | Datatype |
|---|---|
| Empno | Number |
| Name | Text |
| Sal | Currency |
| Address | Text |

**Bonus**

| Field Name | Datatype |
|---|---|
| Empno | Number |
| Monthid | Date |
| Bonus | Number |

a) Use relational ships window to establish a relationship between the Employee and Bonus tables where Employee is the parent table. The relationship should have referential integrity enforced and enter five records in both tables.

b) Sort the records by name alphabetically.

## Exercise # 6

**Create a table product with the following fields:**

| Field Name | Datatype |
|---|---|
| Productno | number |
| Name | text |
| Price | number |
| Expdate | date |
| Description | text |

a) Create primary key and insert five records.

b) Create a query to display only name and price using query wizard.

c) Freeze the field productno

## Exercise # 7

**Create a table Item with the following fields:**

| Field Name | Datatype |
|---|---|
| Itemcode | number |
| Itemname | text |
| Quantity | number |
| Rate | number |

a) Create a primary key and insert five records.
b) Create a query in design view to display a calculated field amount (amount=quantity*rate)

## Exercise # 8

**Create a table projects with the following fields**

| Field Name | Datatype |
|---|---|
| Projectid | number |
| stdname | text |
| Address | text |
| Phone | text |
| Email | text |
| Admission date | date |

a) Assign a primary key
b) Create a tabular form to insert five records in the table using industrial style.
c) Create two buttons to add and delete records.

## Exercise # 9

**Create a table Items with following fields:**

| Field Name | Datatype |
|---|---|
| Itemno | number |
| Itemname | text |
| Description | text |
| Price | number |
| Expdate | date |

a) Assign a primary key
b) Make sure that expdate is greater than current date
c) Create columnar form to insert five records using sandstone style.
d) Insert a command button to close form

## Exercise # 10

**Create table students with following fields:**

| Field Name | Datatype |
|---|---|
| Rollno | number |
| Name | text |
| Class | text |
| Admdate | date |
| Address | text |
| Phone | text |

a) Assign a primary key to table
b) Create a columnar form to input data to insert five records.
c) Make sure that admdate is less than or equal to current date
d) Create two buttons to delete & save records.

## Exercise # 11

**Create the table teacher with suitable primary key:**

| Field Name | Datatype |
|---|---|
| Teacherid | number |
| Name | text |
| Address | text |
| Grade | text |
| Salary | number |

a) Create a form called teacher form in design view to insert five records into table
b) Use "A Grade" as default value in grade field.

## Exercise # 12

**Create a table population with the following fields:**

| Field Name | Datatype |
|---|---|
| Citycode | number |
| Cityname | text |
| Population | number |
| Birthrate | text |
| Educationrate | text |

a) Assign a primary key to table and insert five records.
b) Create a columnar report to display the table contents using report wizard.
c) Set the report title as country population.
d) Use Faisalabad as default value in cityname

## Exercise # 13

**Create a table net cafe with the following fields:**

| Field Name | Datatype |
|---|---|
| Packageno | number |
| Description | text |
| Totalhours | number |
| Rate | number |

a) Assign a primary key to insert five records.
c) Create a query in design view to display a calculated field Total amount (Total amount=Totalhours*Rate)
b) Create a tabular report to display the records in descending order by package no using formal style.
c) Set the report title as package List

## Exercise # 14

**Create a table Guests using table wizard with the following fields:**

| Field Name | Datatype |
|---|---|
| Guestno | autonumber |
| Name | text |
| Address | text |
| Phone | text |
| Email | text |
| Bookingdate | date |

a) Create a primary key

a) Set the required field properties yes for name.

b) Use the short date format for bookingdate (short date)

c) Insert five records in the table

d) Hide the field email.

## Exercise # 15

**Create the table writers using table wizard with following fields**

| Field Name | Datatype |
|---|---|
| Writeno | autonumber |
| Name | text |
| Nationality | text |
| Birthday | date |
| Placeofbirth | text |

a) Use medium date format for birth date.(19-jun-1994)

b) Create a query to display writer name, Nationality and placeofbirth.

c) Assign a primary key to the table and insert five records in the table

d) Use flat option of cell effects.

## Exercise # 16

**Create the following two tables:**

**Customers**

| Field Name | Datatype |
|---|---|
| Ccode | text |
| Cname | text |
| Caddress | text |
| Cphone | text |
| Faxno | text |

**orders**

| FieldName | Datatype |
|---|---|
| ccode | text |
| orderno | number |
| orderdate | date/time |

a) Using Input mask, Set the faxno is in the following format:(92) 041-733288

b) Create a relationship between these tables

c) Enter five records in both tables.

## Exercise # 17

**Create a table called club members with following fields:**

| FieldName | Datatype |
|---|---|
| Socialsecuritynumber | text |
| Name | text |
| Address | text |
| Membershipdate | date/time |
| City | text |
| Homephone | ext |
| Workphone | text |

a) Set the field size 20 for name and address for 80

b) Set primary key

c) Use long date format for membershipdate.(Sunday, June 19,1994)

e) Using Input mask, Set socialsecuritynumber is in the following format: 333-33-33333

d) Enter five records in the table.

## Exercise # 18

**Create a table population with the following fields**

| FieldName | Datatype |
|---|---|
| Citycode | number |
| Cityname | text |
| Population | number |
| Educationrate | text |

a) Assign a primary key to table and insert five records.

b) Using report wizard, create a report to display the population of all cities using corporate style.

c) Set the report title as country population.

# Microsoft Access Projects

## Project 1 | Library Management System

### Introduction

#### Overview Statement

The purpose of this project is to develop *Library Management System* in MS Access.

#### Project Goals

The main goals of the project are as follows:

- Provide a user-friendly interface
- Manage and access books according to authors
- Manage and access books according to categories
- Manage and access library members information
- Manage and access staff members information
- Manage and access book issues, returns & fines, etc

#### Tables in the Project

Database contains the following tables:

- **Staff Personals:** This table stores information about each staff member working as an employee in the library. All the members are allowed to issue books to the members of the library.
- **Authors Data:** This table stores some information about the authors of books.
- **Book Catalogs:** This table stores information about each category of books according to which books are arranged in the racks.
- **Library Members:** This table stores information about the registered members of library.
- **Books Collection:** This table stores basic information about all the books present in the library.
- **Book Issues:** This table stores information about the issuance of different books.

#### Queries in the Project

Database contains the following queries:

- **Author wise Books:** This query provides information about all the available books according to authors.
- **Category wise Books:** This query provides information about all the available books according to categories.
- **Book issues by Authors:** This query provides information about the issue of books according to their authors.
- **Book issues by Category:** This query provides information about the issuance of books according to their categories.
- **Member issued books:** This query provides information about the members and the books issued to them.

- **Books issued by Staff:** This query provides information about the staff and the books which they issued.
- **BookIssues after 30th June 2004:** This query provides information about all the books that are issued after 30th June 2004.
- **Books by desired Author:** This query is a parameter query that prompts user to enter author's name. It then displays book records according to that author.
- **Books in desired Category:** This query is also a parameter query that prompts user to enter a category's name. It then displays book records according to that category.
- **Book issues b/w desired Period:** This query is also a parameter query that prompts user to enter a starting issue date and ending issue date. It then displays books issued between the starting and ending dates.

## Form Objects

Database contains the following forms:

- **Staffs Form:** This form allows inputting and modifying data of staff members working as an employee in the library.
- **Authors Form:** This form allows inputting and modifying data about the authors of the books.
- **BookCatalogs Form:** This form allows inputting and modifying data about categories of books.
- **LibMembers Form:** This form allows inputting and modifying data about the registered members of library.
- **BooksCollection Form:** This form allows inputting and modifying data about the library books.
- **BookIssues Form:** This form allows inputting and modifying data about issuance of books to different members.
- **Options Window:** This form is designed to provide quick access to each object in the database. It contains buttons that refer to some table, query, form, report in the database. It is similar to a Switchboard.
- **Welcome Window:** This form is designed to provide an interface to the database. It is used to choose whether to navigate the database or simply close MS Access.

## Report Objects

Database contains the following reports:

- **Author wise Books:** This report provides preview of information about all the available books according to authors.
- **Category wise Books:** This report provides preview of information about all the available books according to categories.
- **Book issues by Authors:** This report provides preview of information about the issued books according to their authors.
- **Books issued to Members:** This report provides preview of information about the members and the books issued to them.
- **Books by desired Author:** This report is a parameter query and it prompts user to enter author's name. It then generates a report of books of author entered by the user.
- **Book issues b/w desired Period:** This report is a parameter query and it prompts the user to enter starting date and ending date. It then generates a report of books issued between the entered dates.

# 1. Database Creation

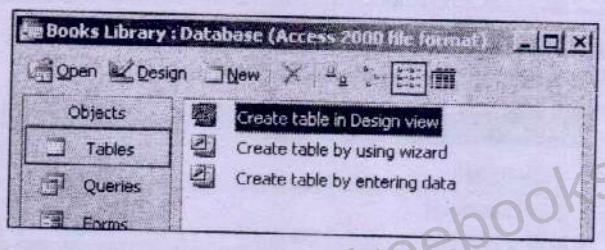
## Creating a New Blank Database

1. Click **Start > Programs > Microsoft Access** to start MS Access.
2. Select **Blank Access Database** option.
3. Click **OK**. A dialog box will appear to input database name.
4. Type 'Books Library' in **File Name** box.
5. Select **My Documents** folder to save the database.
6. Click **Create**. A new database will be created and the **Database** window will appear.

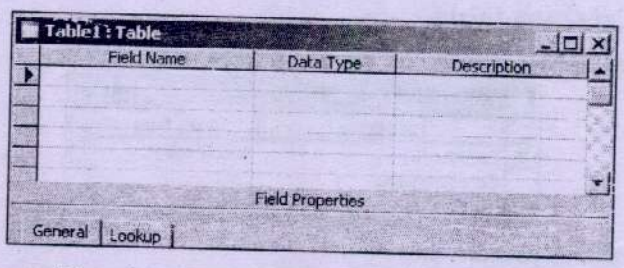
# 2. Tables Creation

## Creating Staff Personals table

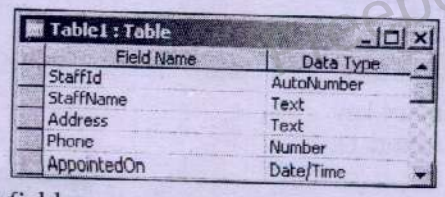
1. Click on the **Tables** object in the **database** window.



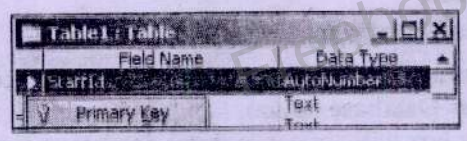

2. Double click **Create table in Design view** in **Database** window. A new blank table will appear.



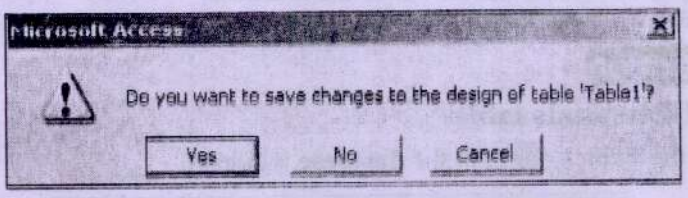

3. Type the name of first field as **StaffId** in **Field Name** column
4. Press **Tab** key to move to **Data Type** Column.
5. Click drop down arrow and select data type **AutoNumber** for the field.
6. Press **Tab** key to move to **Description** column and type any note for the field if you want.
7. Press Tab key to move to **Field Name** for the next field.
8. Repeat step 3 to 8 until all fields have been defined.




9. Click on the **StaffId** field.
10. Click on **Primary key** icon on standard toolbar. The key symbol will appear in the selected field.




11. Click on the Close button ✕ on Table Design view window. The close dialog box will appear.

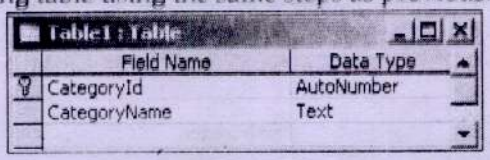



12. Click **Yes** to save the table. The **Save As** dialog box will appear.
13. Type 'Staff Personals' in the **Table Name** text box.


14. Click **OK** button to save the table with the typed name.

## Creating Book Catalogs table

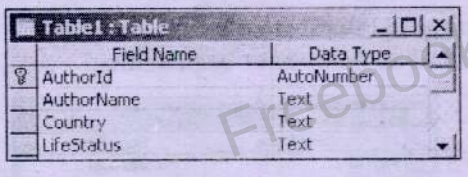
1. Create the following table using the same steps as previous:




2. Set **CategoryId** as primary key.
3. Save the table as 'Book Catalogs'.
4. Close the table.

## Creating Authors Data table

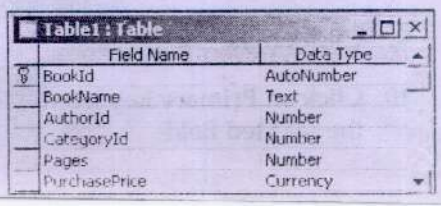
1. Create the following table using the same steps as previous:




2. Set **AuthorId** as primary key.
3. Save the table as 'Authors Data'.
4. Close the table.

## Creating Books Collection table

1. Create the following table using the same steps as previous:
2. Set **BookId** as primary key.



3. Save the table as 'Books Collection'.
4. Close the table.

## Creating Library Members table

1. Create the following table using the same steps as previous:
2. Set **MemId** as primary key.
3. Save the table as 'Library Members'.
4. Close the table.

Table1 : Table

| Field Name | Data Type |
|---|---|
| MemId | AutoNumber |
| MemName | Text |
| MembershipDate | Date/Time |
| Address | Text |
| Phone | Number |

## Creating Book Issues table

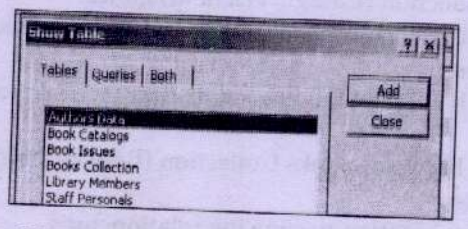
1. Create a table with the following fields and data types:
2. Set **IssueNo** as primary key.
3. Set the **Caption** property (under the **General** tab) of **StaffId** field as **IssuedBy**.
4. Save the table as 'Book Issues'.
5. Close the table.

Table1 : Table

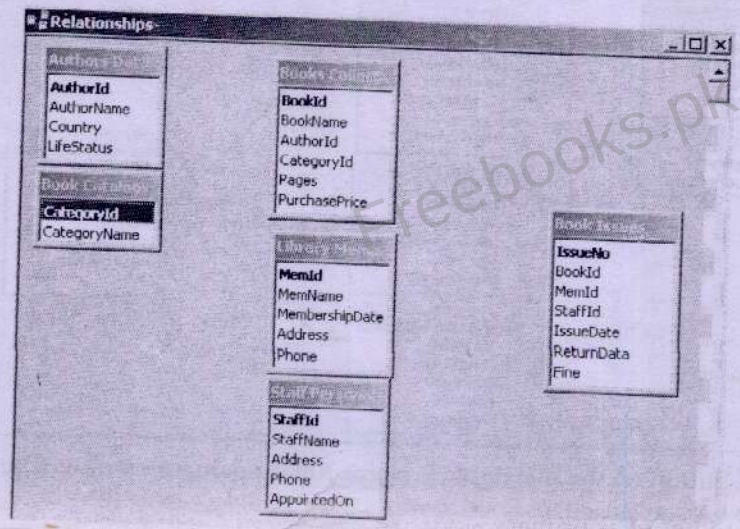
| Field Name | Data Type |
|---|---|
| IssueNo | AutoNumber |
| BookId | Number |
| MemId | Number |
| StaffId | Number |
| IssueDate | Date/Time |
| ReturnDate | Date/Time |
| Fine | Currency |

## 3. Defining Relationships

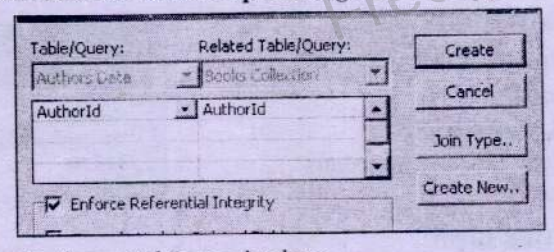
1. Click **Tools > Relationship** OR Click **Relationship** icon on the **Standard** toolbar of access window. The **Relationship** window with **Show Table** dialog box will appear.



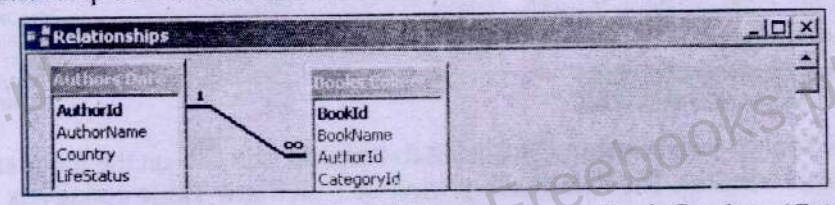

2. Double click every table in the list box one by one to add in **Relationships** window.
3. Click **Close** button after adding the tables. All the added tables appear as follows in **Relationships** window.



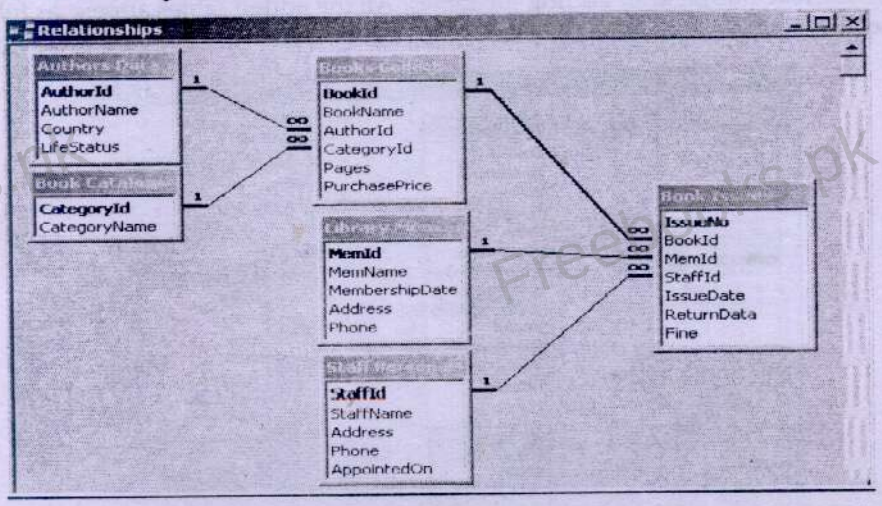
4. Drag 'AuthorId' field from **Authors Data** table to 'AuthorId' field in **Books Collection** table. An **Edit Relationships** dialog box will appear.




5. Check the **Enforce Referential Integrity** box.
6. Click **Create** button on **Edit Relationships** dialog box.
7. The relationship will be created. A line joining two related fields in the tables will appear in the Relationship window. This line also indicates a **One-to-Many** relationship between the linked tables:




8. Use the same procedure to create relationship among Book Catalogs (**CategoryId** field) and Books Collection (**CategoryId** field) tables.
9. Create relationship between Library Members (**MemId** field) and Book Issues (**MemId** field) tables.
10. Create relationship between Staff personals (**StaffId** field) and Book Issues (**StaffId** field) tables.
11. Create relationship between Books Collection (**BookId** field) and Book Issues (**BookId** field) tables.
12. Press **Ctrl+S** key combination to save the relationships.
13. The Relationships window looks like as follows after creating all above relationships.

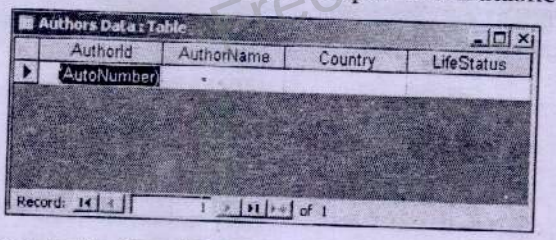



14. Click ✕ button on the top right corner of **Relationships** window. The **Relationships** window will be closed.

# 4. Adding Records

## Adding Records in Authors Data Table

1. Double Click on the table 'Authors Data' to open it in **Datasheet** view.

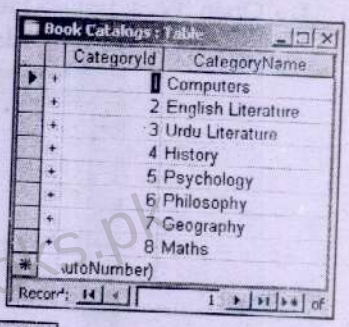



2. By default cursor is in the first field, press **Tab** or **Enter** key to move to next field. There is no need to write anything in **AutoNumber** fields.
3. Type 'Tariq Mahmood' in **AuthorName** field.
4. Press **Tab** key to move forward.
5. In the same way fill all the other fields.
6. Repeat steps 3 to 5 for adding remaining records, as shown in the figure below:


7. Press **Ctrl + F4** to close the opened table.

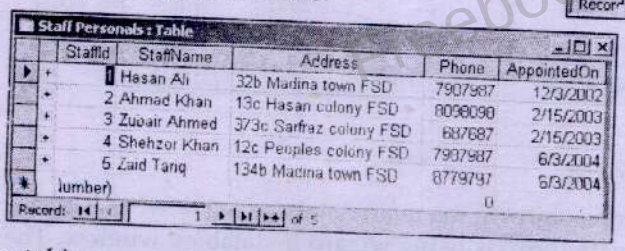
## Adding Records to Book Catalogs Table

1. Open 'Book Catalogs' table in datasheet view.
2. Add following records in the table:
3. Close the table.




## Adding records to Staff Personals Table

1. Open 'Staff Personals' table in datasheet view.
2. Add following records in the table:




3. Close the table.

## Adding records to Library Members Table

1. Open 'Library Members' table in datasheet view.
2. Add following records in the table:

Library Members : Table

| MemId | MemName | MembershipDate | Address | Phone |
|---|---|---|---|---|
| 1 | Asad Ali | 12/20/2002 | 13b Gulistan colony FSD | 432141 |
| 2 | Saad Shah | 12/20/2002 | 40b Poeples colonyFSD | 765742 |
| 3 | Hasan Khan | 1/1/2003 | 98c Madina town FSD | 53454 |
| 4 | Maaz Ahmed | 1/3/2003 | 37p Sarfraz colony FSD | 64356 |
| 5 | Zubair Tariq | 1/30/2003 | 56a Peoples colony FSD | 645645 |
| 6 | Zaheer Ali | 3/3/2003 | 123a Garaznwala road FSD | 654746 |
| 7 | Nauman Ahmed | 3/30/2003 | 14k Gulistan colony FSD | 765757 |
| 8 | Usman Shah | 6/13/2003 | 12q Sheikh colony FSD | 765755 |
| 9 | Ali Ahmed | 2/8/2004 | 198d Nayyab colony FSD | 36656 |
| 10 | Talha Khan | 2/9/2004 | 64b Gulistan colony FSD | 36436 |
| Number) | | | | 0 |

Record: 1 of 10

3. Close the table.

## Adding records to Books Collection Table

1. Open 'Books Collection' table in datasheet view.
2. Add following records in the table:

Books Collection : Table

| BookId | BookName | AuthorId | CategoryId | Pages | PurchasePrice |
|---|---|---|---|---|---|
| 1 | Fundamental study of Databases | 2 | 1 | 250 | Rs. 150.00 |
| 2 | Visual Programming using Visual Basic | 1 | 1 | 320 | Rs. 190.00 |
| 3 | Data Communication | 13 | 1 | 200 | Rs. 120.00 |
| 4 | Colors of Silence | 4 | 2 | 150 | Rs. 100.00 |
| 5 | Philosophy Basic Readings | 9 | 6 | 250 | Rs. 80.00 |
| 6 | The Seeing Eye | 10 | 3 | 180 | Rs. 60.00 |
| 7 | Antony & Cleopatra | 6 | 2 | 130 | Rs. 100.00 |
| 8 | Romeo & Juliet | 6 | 2 | 80 | Rs. 70.00 |
| 9 | The story of Philosophy | 5 | 6 | 120 | Rs. 90.00 |
| 10 | The pleasures of Philosophy | 7 | 6 | 230 | Rs. 190.00 |

3. Close the table.

## Adding records to Book Issues Table

1. Open 'Book Issues' table in datasheet view.
2. Add following records in the table:

| IssueNo | BookId | MemId | IssuedBy | IssueDate | ReturnData | Fine |
|---|---|---|---|---|---|---|
| 1 | 6 | 2 | 3 | 2/3/2004 | 2/9/2004 | Rs. 0.00 |
| 2 | 1 | 4 | 2 | 2/3/2004 | 2/20/2004 | Rs. 10.00 |
| 3 | 1 | 2 | 2 | 3/3/2004 | 3/8/2004 | Rs. 0.00 |
| 4 | 4 | 5 | 1 | 3/12/2004 | 3/30/2004 | Rs. 15.00 |
| 5 | 8 | 4 | 3 | 3/19/2004 | 3/25/2004 | Rs. 0.00 |
| 6 | 3 | 5 | 3 | 4/1/2004 | 4/8/2004 | Rs. 0.00 |
| 7 | 2 | 6 | 1 | 4/3/2004 | 4/13/2004 | Rs. 0.00 |
| 8 | 9 | 5 | 3 | 6/2/2004 | 6/18/2004 | Rs. 2.00 |
| 9 | 6 | 3 | 5 | 6/30/2004 | 7/8/2004 | Rs. 0.00 |
| 10 | 4 | 5 | 2 | 8/10/2004 | 8/19/2004 | Rs. 0.00 |
| oNumber) | 0 | 0 | 0 | | | Rs. 0.00 |

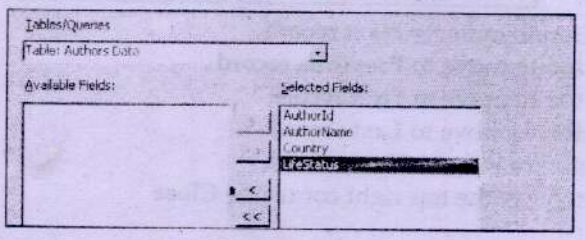
Record: 1 of 10

3. Close the table.

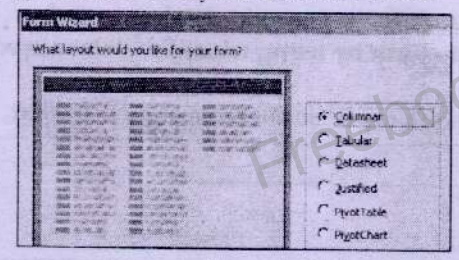
## Creating Authors Form

1. Click on the **Forms** object in the main **database** window.
2. Double click **Create form by using wizard** in database window. The **Form Wizard** will appear.

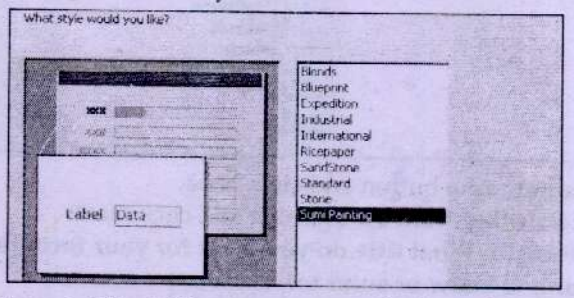

3. Click **Tables/Queries** drop-down list and choose *Authors Data* table. All fields of the selected table will appear in the **Available Fields** box.
4. Click **double-right arrow** button >> . All fields will move to the **Selected Fields** list.



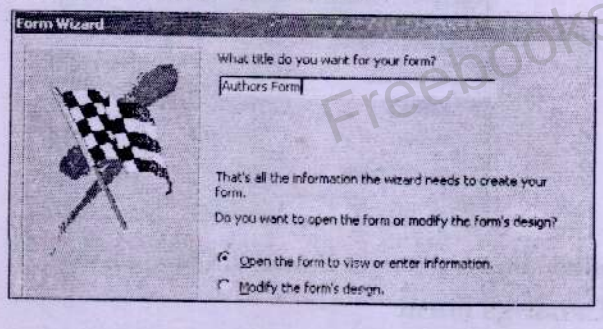

5. Click Next.
6. Select **Columnar** radio button for layout and click **Next**.



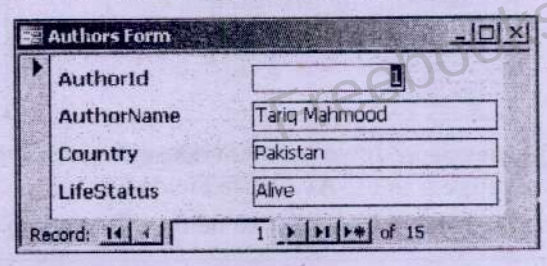

7. Select **Sumi Painting** from the style list and click **Next**.




8. Type *Authors Form* in **What title do you want for your form?** text box and select the **Open the form to show or enter information.**



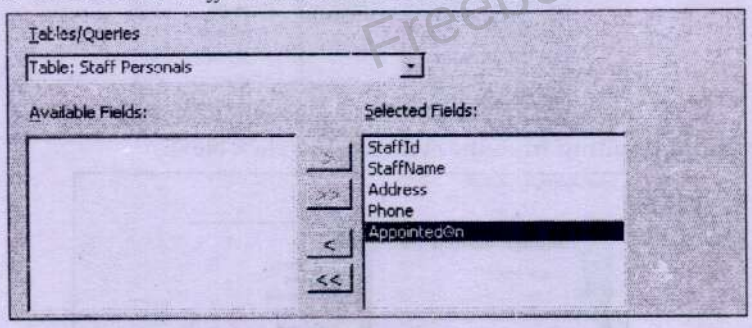
9. Click **Finish.** The form will be displayed in **Form view.**



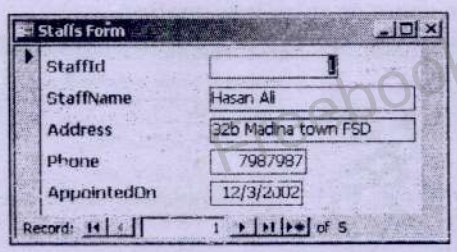

10. Press button to move to **Next record.**
11. Press button to move to **Previous record.**
12. Press button to move to **First record.**
13. Press button to move to **Last record.**
14. Press button to insert a **New record.**
15. Click button (on the top right corner) to **Close** the form.

## Creating Staffs Form

1. Click on the **Forms** object in the main **database** window.
2. Double click **Create form by using wizard** in database window. The **Form Wizard** will appear.
3. Select all the fields of *Staff Personals* table and click **Next.**




4. Select **Columnar** radio button and click **Next.**
5. Select **Sumi Painting** from the style list and click **Next.**
6. Type *Staffs Form* in **What title do you want for your form?** text box and select the **Open the form to show or enter information.**
7. Click **Finish.** The form will be displayed in **Form view.**

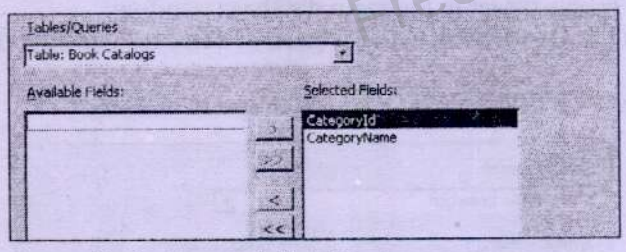



8. Click button (on the top right corner) to **Close** the form.

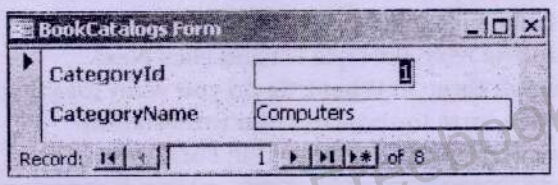
## Creating BookCatalogs Form

1. Click on the **Forms** object in the main **database** window.

2. Double click **Create form by using wizard** in database window. The **Form Wizard** will appear.
3. Select all the fields of *Book Catalogs table* and click **Next**.



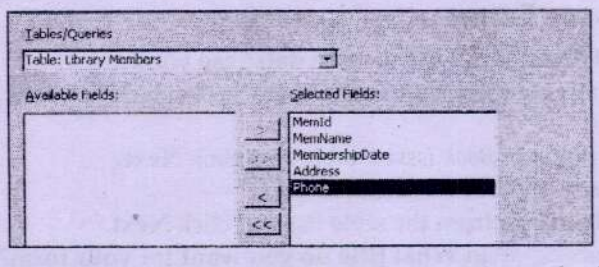

4. Select **Columnar** radio button and click **Next**.
5. Select **Sumi Painting** from the style list and press **Next**.
6. Type *BookCatalogs Form* in **What title do you want for your form?** text box and select the **Open the form to show or enter information**.
7. Click **Finish.** The form will be displayed in **Form view.**



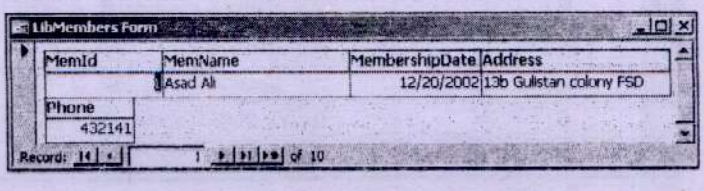

8. Click ✕ button (on the top right corner) to **Close** the form.

## Creating LibMembers Form

1. Click on the **Forms** object in the main **database** window.
2. Double click **Create form by using wizard** in database window. The **Form Wizard** will appear.
3. Select all the fields of *Library Members table* and click **Next**.



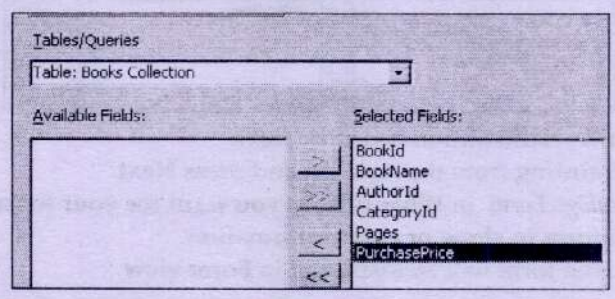

4. Select **Justified** radio button and click **Next**.
5. Select **Sumi Painting** from the style list and click **Next**.
6. Type *LibMembers Form* in **What title do you want for your form?** text box and select the **Open the form to show or enter information**.
7. Click **Finish.** The form will be displayed in **Form view.**



8. Click ✕ button (on the top right corner) to **Close** the form.

## Creating BooksCollection Form

1. Click on the **Forms** object in the main **database** window.
2. Double click **Create form by using wizard** in database window. The **Form Wizard** will appear.
3. Select all the fields of *Books Collection* table and click **Next**.




4. Select **Tabular** radio button and click **Next**.
5. Select **Sumi Painting** from the style list and click **Next**.
6. Type *BooksCollection Form* in **What title do you want for your form?** text box and select the **Open the form to show or enter information**.
7. Click **Finish.** The form will be displayed in **Form view.**

**BooksCollection Form**

| BookId | BookName | AuthorId | CategoryId | Pages | PurchasePrice |
|---|---|---|---|---|---|
| 1 | Fundamental study of Databases | 2 | 1 | 250 | Rs. 150.00 |
| 2 | Visual Programming using Visual Basic | 1 | 1 | 320 | Rs. 190.00 |
| 3 | Data Communication | 13 | 1 | 200 | Rs. 120.00 |

8. Click ✕ button (on the top right corner) to **Close** the form.

## Creating BookIssues Form

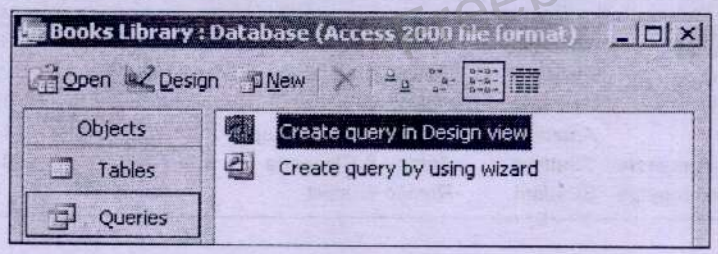
1. Click on the **Forms** object in the main **database** window.
2. Double click **Create form by using wizard** in database window. The **Form Wizard** will appear.
3. Select all the fields of *Book Issues* table and click **Next**.
4. Select **Tabular** radio button and click **Next**.
5. Select **Sumi Painting** from the style list and click **Next**.
6. Type *BookIssues Form* in **What title do you want for your form?** text box and select the **Open the form to show or enter information**.
7. Click **Finish.** The form will be displayed in **Form view.**

**BookIssues Form**

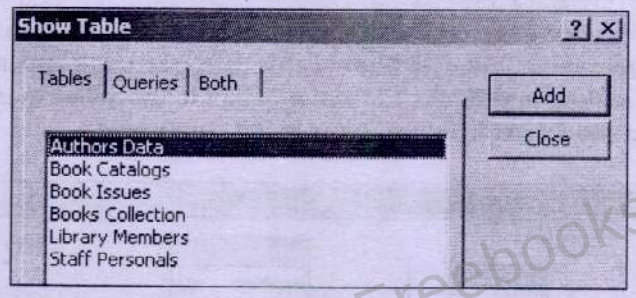
| IssueNo | BookId | MemId | IssuedBy | IssueDate | ReturnData | Fine |
|---|---|---|---|---|---|---|
| 1 | 6 | 2 | 3 | 2/3/2004 | 2/9/2004 | 0.00 |
| 2 | 1 | 4 | 2 | 2/3/2004 | 2/20/2004 | 10.00 |
| 3 | 1 | 2 | 2 | 3/3/2004 | 3/8/2004 | 0.00 |
| 4 | 4 | 5 | 1 | 3/12/2004 | 3/30/2004 | 15.00 |
| 5 | 8 | 4 | 3 | 3/19/2004 | 3/25/2004 | 0.00 |

## Creating query to display Author wise Books

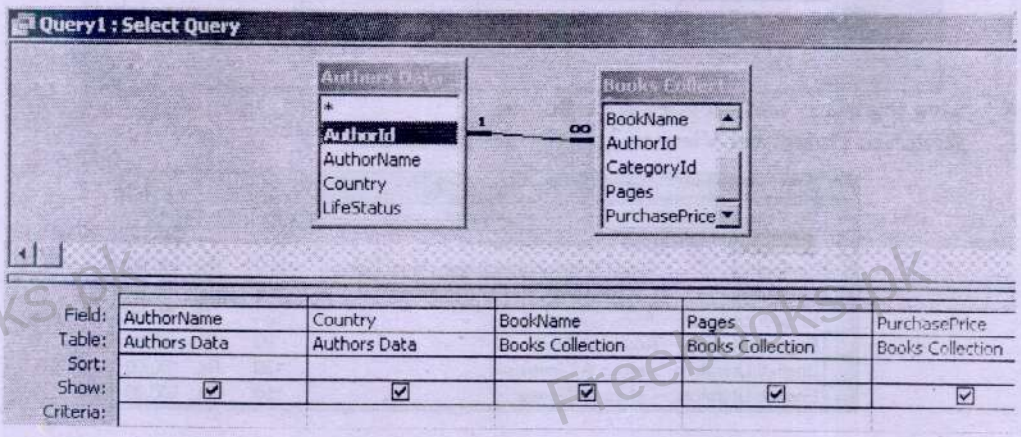
1. Click on **Queries** object in the main database window.



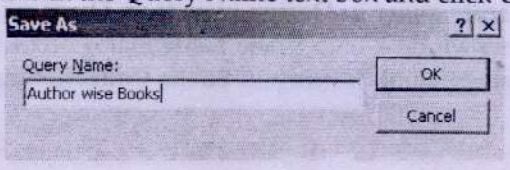

2. Double-click **Create query in Design view**. The **Select Query** window with Show **Table** dialog box will appear.




3. Select the table 'Authors Data' table.
4. Click **Add** button. The table will be added to the **Select Query** window.
5. Click **Close** button to close the dialog box.
6. Double-click on the first field AuthorName in the Table. The field will be added to the **Field** row in the **design grid**.
7. Add other fields to the query.




8. Press **Ctrl+S** to save the query. The **Save As** dialog box will appear.
9. Type *Author wise Books* in the **Query Name** text box and click **OK**.



10. Switch to **Datasheet View** of query to see the results.

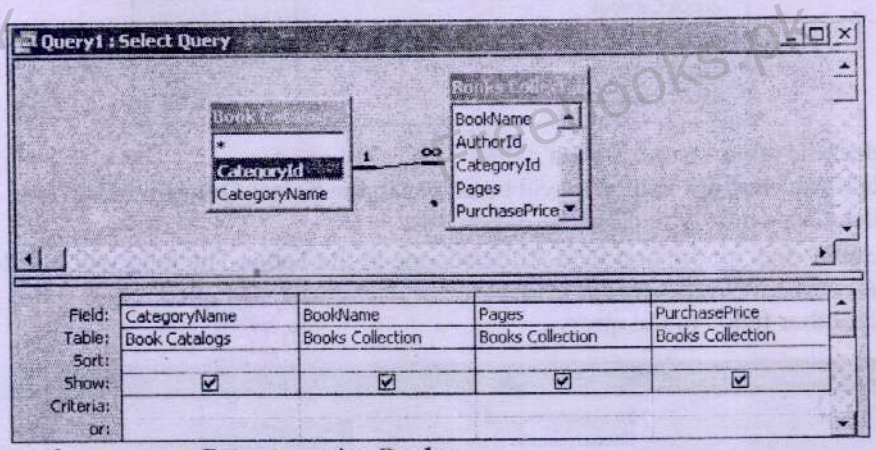
Author wise Books : Select Query

| AuthorName | Country | BookName | Pages | PurchasePrice |
|---|---|---|---|---|
| Tariq Mahmood | Pakistan | Visual Programming using Visual Basic | 320 | Rs. 190.00 |
| Imran saeed | Pakistan | Fundamental study of Databases | 250 | Rs. 150.00 |
| Percy Shelly | Germany | Colors of Silence | 150 | Rs. 100.00 |
| Will Durant | America | The story of Philosophy | 120 | Rs. 90.00 |
| William Shakespeare | Stratford | Antony & Cleopatra | 130 | Rs. 100.00 |
| William Shakespeare | Stratford | Romeo & Juliet | 80 | Rs. 70.00 |

11. Click ☒ button (on the top right corner of **Select Query** title bar) to close the query.

## Creating query to display Category wise Books

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.
2. Add **Book Catalogs** and **Books Collection** tables to the **Select query** window.
3. Add fields as shown in the figure below to the **Query grid**.




4. Save the query as **Category wise Books**.
5. Switch to **Datasheet View** of query to see the results.

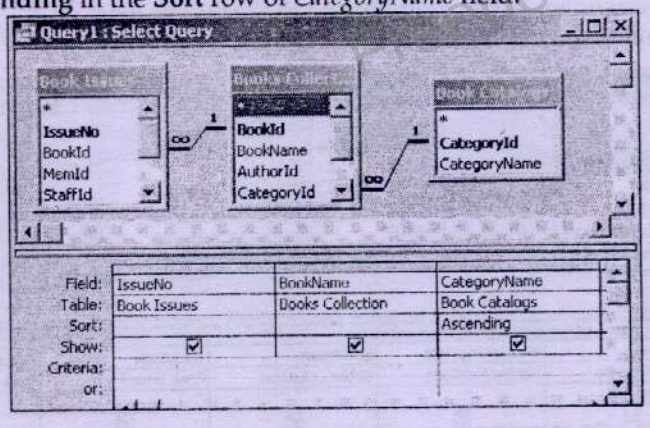
Category wise Books : Select Query

| CategoryName | BookName | Pages | PurchasePrice |
|---|---|---|---|
| Computers | Data Communication | 200 | Rs. 120.00 |
| Computers | Visual Programming using Visual Basic | 320 | Rs. 190.00 |
| Computers | Fundamental study of Databases | 250 | Rs. 150.00 |
| English Literature | Julius Caesar | 105 | Rs. 90.00 |
| English Literature | Romeo & Juliet | 80 | Rs. 70.00 |
| English Literature | Antony & Cleopatra | 130 | Rs. 100.00 |
| English Literature | Colors of Silence | 150 | Rs. 100.00 |

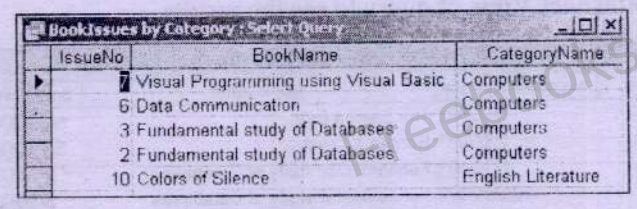
## Creating query to display BookIssues by Category

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.
2. Add **Book Catalogs, Book Issues** and **Books Collection** tables to the **Select query window.**

3. Add fields as shown in the figure below to the **Query grid**.
4. Select **Ascending** in the **Sort** row of *CategoryName* field.

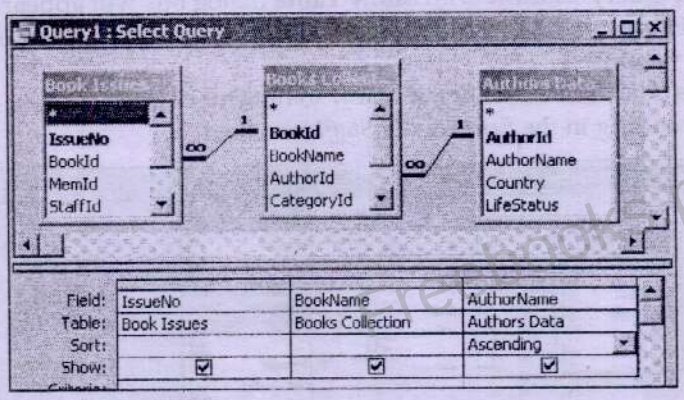



5. Save the query as **BookIssues by Category**.
6. Switch to **Datasheet View** of query to see the results.




## Creating query to display BookIssues by Authors

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.
2. Add **Authors Data**, **Book Issues** and **Books Collection** tables to the query window.
3. Add fields as shown in the figure below to the **Query grid**.
4. Select **Ascending** in the **Sort** row of *AuthorName* field.

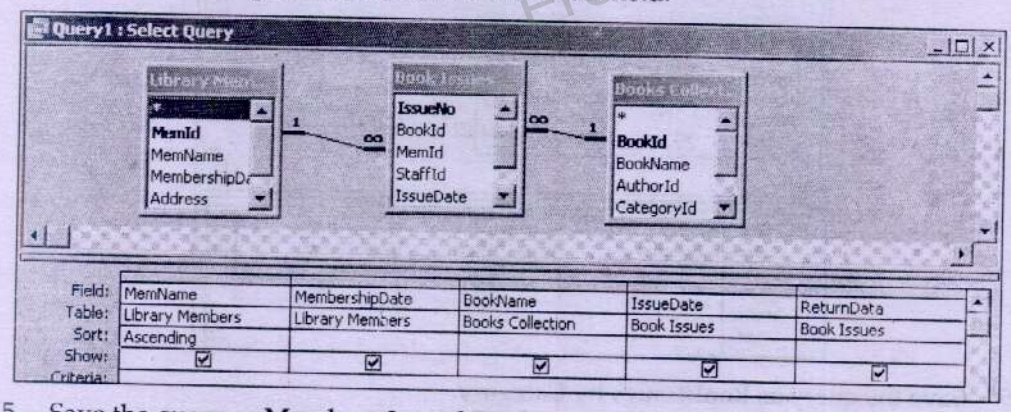



5. Save the query as **BookIssues by Authors**.
6. Switch to **Datasheet View** of query to see the results.

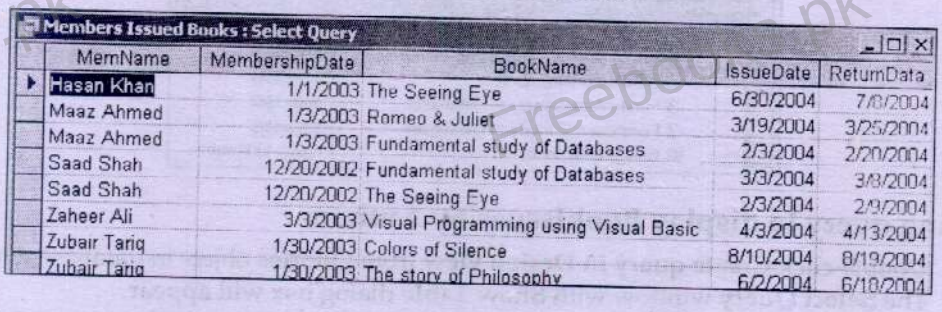
## Creating query to display MembersIssued Books

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.

2. Add **Library Members**, **Book Issues** and **Books Collection** tables to the query window.
3. Add fields as shown in the figure below to the **Query grid**.
4. Select **Ascending** in the **Sort** row of *MemName* field.




5. Save the query as **Members Issued Books**.
6. Switch to **Datasheet View** of query to see the results.

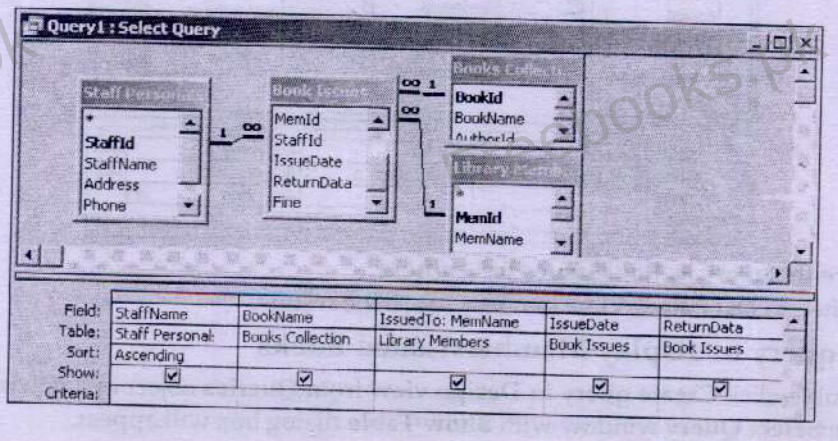


Members Issued Books : Select Query

| MemName | MembershipDate | BookName | IssueDate | ReturnData |
|---|---|---|---|---|
| Hasan Khan | 1/1/2003 | The Seeing Eye | 6/30/2004 | 7/0/2004 |
| Maaz Ahmed | 1/3/2003 | Romeo & Juliet | 3/19/2004 | 3/25/2004 |
| Maaz Ahmed | 1/3/2003 | Fundamental study of Databases | 2/3/2004 | 2/20/2004 |
| Saad Shah | 12/20/2002 | Fundamental study of Databases | 3/3/2004 | 3/8/2004 |
| Saad Shah | 12/20/2002 | The Seeing Eye | 2/3/2004 | 2/9/2004 |
| Zaheer Ali | 3/3/2003 | Visual Programming using Visual Basic | 4/3/2004 | 4/13/2004 |
| Zubair Tariq | 1/30/2003 | Colors of Silence | 8/10/2004 | 8/19/2004 |
| Zubair Tariq | 1/30/2003 | The story of Philosophy | 6/2/2004 | 6/10/2004 |

## Creating query to display Books issued by Staff

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.
2. Add **Staff Personals**, **Library Members**, **Book Issues** and **Books Collection** tables to the query window.
3. Add fields as shown in the figure below to the **Query grid**.
4. Select **Ascending** in the **Sort** row of *StaffName* field.



5. Save the query as **Books issued by Staff**.
6. Switch to Datasheet View of query to see the results.

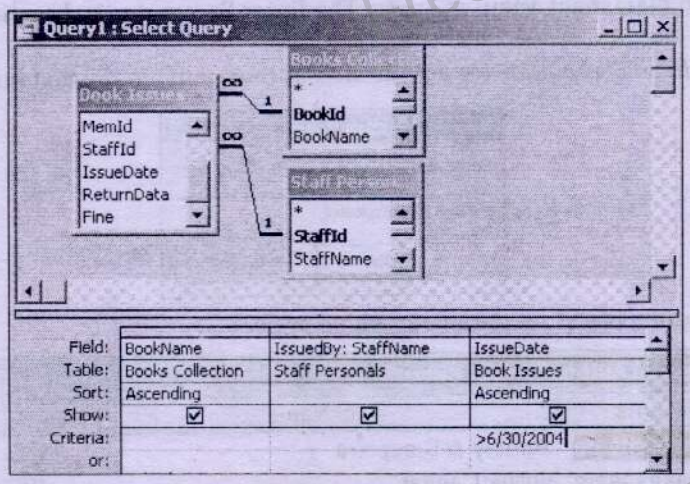
Books issued by Staff : Select Query

| StaffName | BookName | IssuedTo | IssueDate | ReturnData |
|---|---|---|---|---|
| Ahmad Khan | Colors of Silence | Zubair Tariq | 8/10/2004 | 8/19/2004 |
| Ahmad Khan | Fundamental study of Databases | Saad Shah | 3/3/2004 | 3/8/2004 |
| Ahmad Khan | Fundamental study of Databases | Maaz Ahmed | 2/3/2004 | 2/20/2004 |
| Hasan Ali | Visual Programming using Visual Basic | Zaheer Ali | 4/3/2004 | 4/13/2004 |
| Hasan Ali | Colors of Silence | Zubair Tariq | 3/12/2004 | 3/30/2004 |
| Zaid Tariq | The Seeing Eye | Hasan Khan | 6/30/2004 | 7/8/2004 |
| Zubair Ahmed | The story of Philosophy | Zubair Tariq | 6/2/2004 | 6/10/2004 |
| Zubair Ahmed | Data Communication | Zubair Tariq | 4/1/2004 | 4/8/2004 |
| Zubair Ahmed | Romeo & Juliet | Maaz Ahmed | 3/19/2004 | 3/25/2004 |
| Zubair Ahmed | The Seeing Eye | Saad Shah | 2/3/2004 | 2/9/2004 |

Record: 1 of 10

## Creating query to display BookIssues after 30th June 2004

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.
2. Add **Staff Personals, Book Issues** and **Books Collection** tables to the query window.
3. Add fields as shown in the figure below to the **Query grid.**
4. Select **Ascending** in the **Sort** row of *BookName* field.
5. Type *>6/30/2004* in the **Criteria** row of *IssueDate* field.




6. Save the query as **BookIssues after 30th June 2004.**
7. Switch to Datasheet View of query to see the results.

BookIssues after 30th June 2004 : Select Query

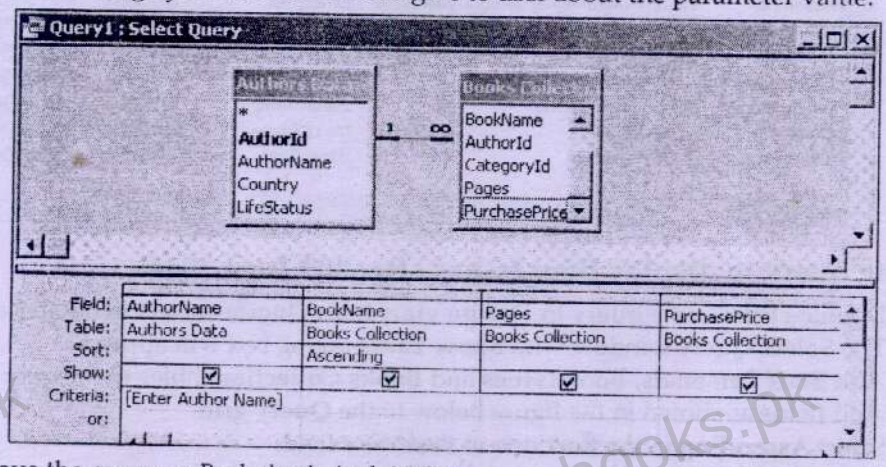
| BookName | IssuedBy | IssueDate |
|---|---|---|
| Colors of Silence | Ahmad Khan | 8/10/2004 |

Record: 1 of 1

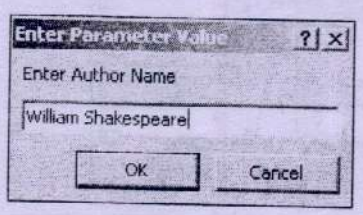
## Creating parameter query to display Books by desired Author

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.

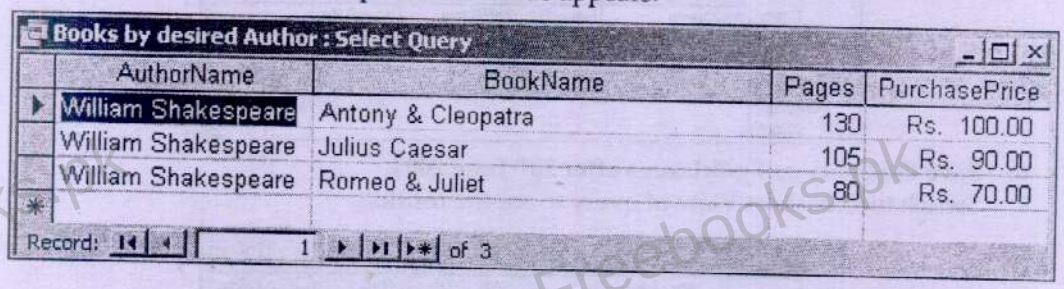
2. Add *Authors Data* & *Books Collection* tables to the query window.
3. Add the fields to the **Query grid** as shown in the figure below to **Query grid**.
4. Select *Ascending* in **Sort** box of *BookName* field.
5. In **Criteria** box of *AuthorName* field, type **[Enter Author Name]**. This text appears in the prompt when you run the query. You may enter any other text depending upon what message you want to want to give to user about the parameter value.




6. Save the query as *Books by desired Author*.
7. Switch to **Datasheet View** of query. The **Enter Parameter Value** dialog box will appear
8. Type *William Shakespeare* (or any desired author name in the text box and click **OK**.



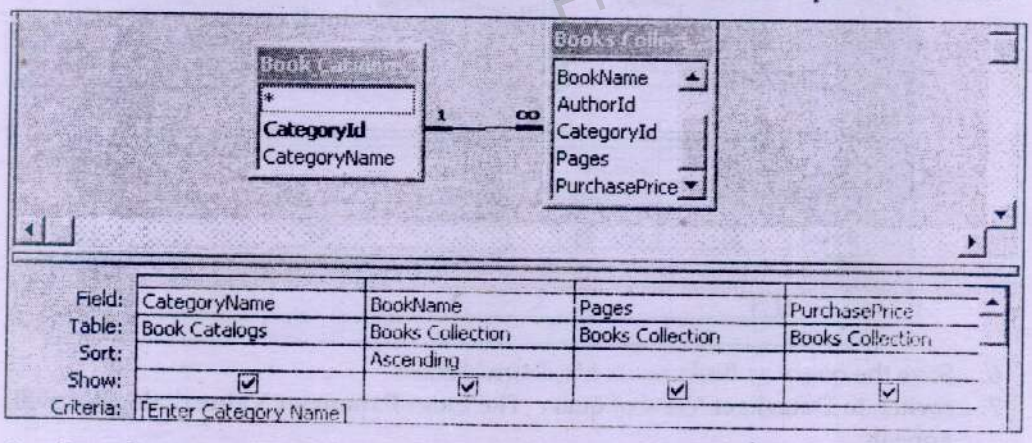

9. Query results based on parameter value appears.



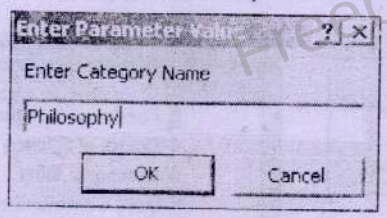

## Creating parameter query to display Books in desired Category

1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.
2. Add *Book Catalogs* & *Books Collection tables* to the query window.
3. Add the fields to the query grid as shown in the figure below.
4. Select *Ascending* in **Sort** box of *BookName* field.

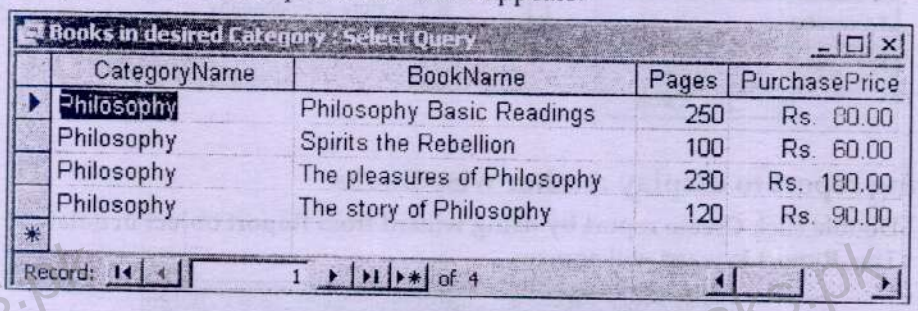
5. In **Criteria** box of *CategoryName* field, type **[Enter Category Name]**. This text appears in the prompt when you run the query. You may enter any other text depending upon what message you want to want to give to user about the parameter value.




6. Save the query as *Books in desired Category*.
7. Switch to **Datasheet View** of query. **Enter Parameter Value** dialog box will appear.
8. Type *Philosophy* (or any desired category name) in the text box and press OK.




9. Query results based on parameter value appears.



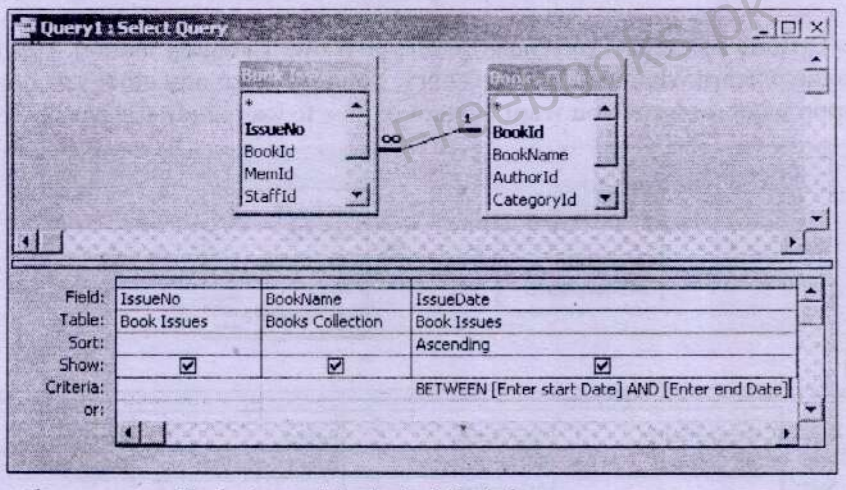
Books in desired Category : Select Query

| CategoryName | BookName | Pages | PurchasePrice |
|---|---|---|---|
| Philosophy | Philosophy Basic Readings | 250 | Rs. 80.00 |
| Philosophy | Spirits the Rebellion | 100 | Rs. 60.00 |
| Philosophy | The pleasures of Philosophy | 230 | Rs. 180.00 |
| Philosophy | The story of Philosophy | 120 | Rs. 90.00 |

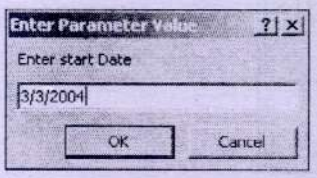
Record: 1 of 4

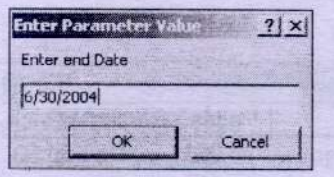
## Creating parameter query to display Books issues b/w desired Period

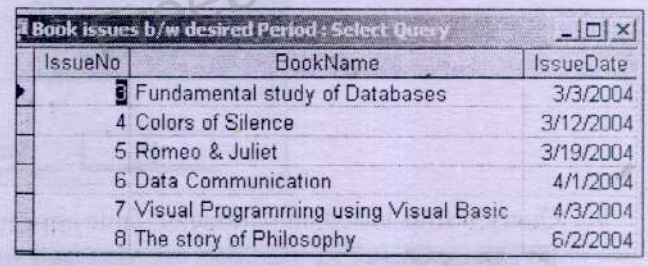
1. Double-click **Create query in Design view** from **Queries** object in database window. The **Select Query** window with **Show Table** dialog box will appear.
2. Add *Book Issues* & *Books Collection tables* to the query window.
3. Add the fields to the query grid as shown in the figure below.
4. Select *Ascending* in **Sort** box of *IssueDate* field.
5. In **Criteria** box of *IssueDate* field, type **BETWEEN [Enter Start Date] AND [Enter end date]**. This text appears in the prompts when you run the query. You may enter any other text depending upon what message you want to want to give to user about the parameter values.




6. Save the query as *Books issues b/w desired Period*.
7. Switch to **Datasheet View** of query. The **Enter Parameter Value** dialog box will appear.
8. Type *3/3/2004* or any desired start date in the text box and click **OK**.
9. Type *6/30/2004* or any desired end date in the text box and click **OK**.
10. Query results based on parameter values appear.

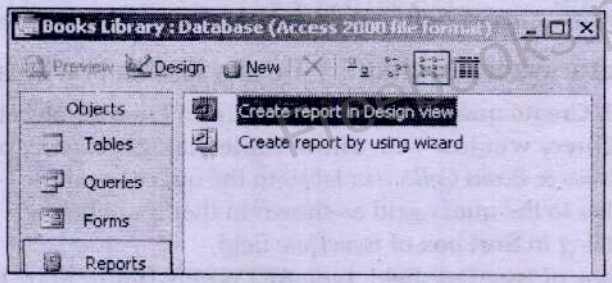


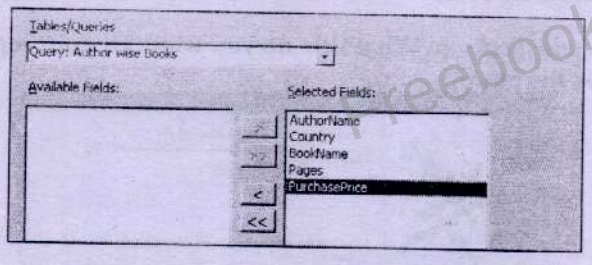




## Creating report to display Author wise Books

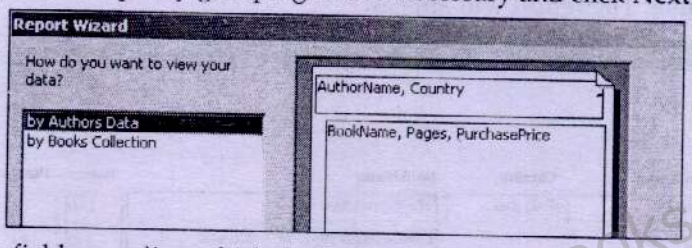
1. Double click **Create report by using wizard** from **Report object** in database window. The **Report Wizard** will appear.



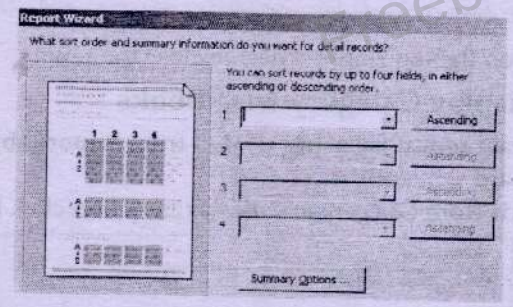

2. Click **Tables/Queries** drop-down list & choose *Author wise Books* query. The fields of the selected query will be displayed in the **Available fields** box.



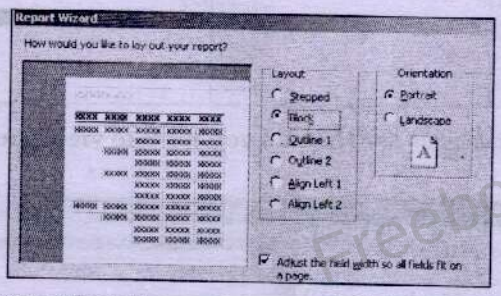

3. Click on double right arrow icon. All field of the query will move to the **Selected Fields** list.
4. Click **Next.**
5. Select any field to specify grouping level if necessary and click **Next**



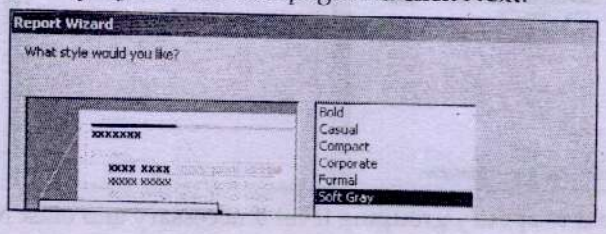

6. Select any field according which data record will be sorted in report.



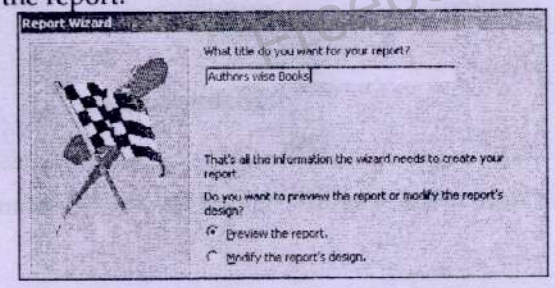

7. Click Next.
8. Select **layout** as **Block** and **orientation** as **Portrait** for the report.
9. Click **Next.**



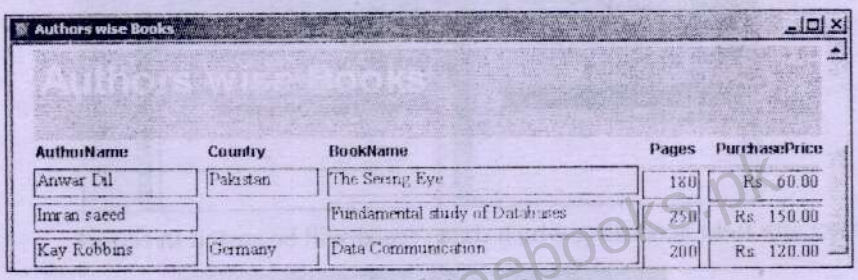

10. Choose the *Soft Gray style* in the next page and click **Next.**



11. Type *Authors wise Books* in **What title do you want for your report?** text box and select preview the report.



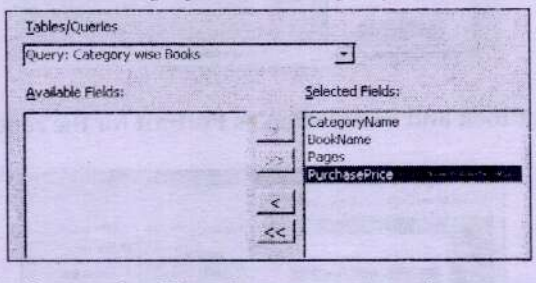

12. Click **Finish.** The report will appear.



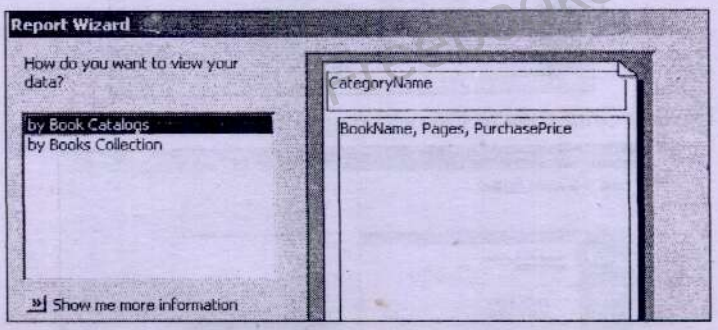

13. Click ✕ button (on the top right corner) to close the report.

## Creating report to display Category wise Books

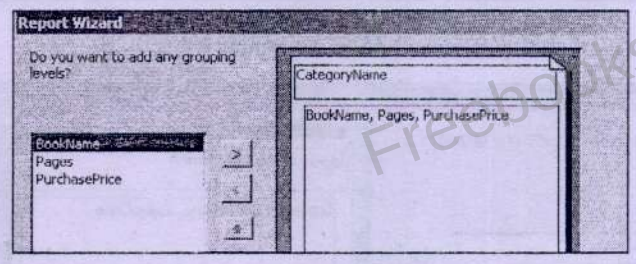
1. Double click **Create report by using wizard** from **Report object** in database window. The **Report Wizard** will appear.
2. Select all the fields from *Category wise Books* query and click **Next**.



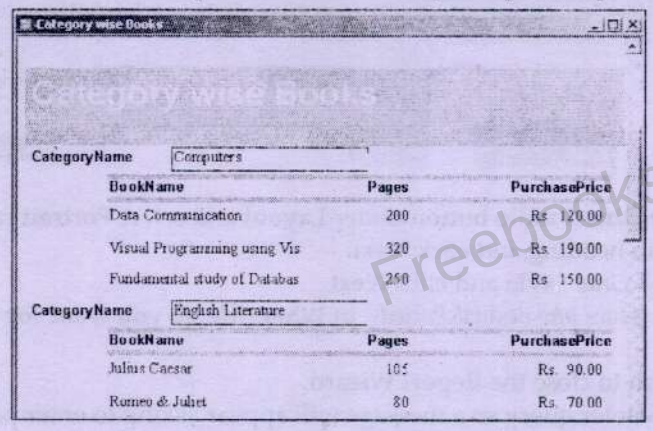

3. Select *by Book Catalogs* under **How do you want to view your data?** title and click **Next**.




4. Select any field to specify grouping level if necessary and click **Next**.



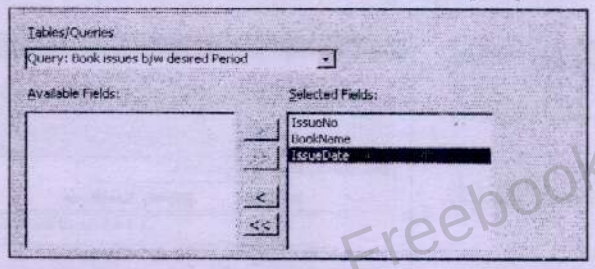

5. Choose **Outline 1** radio button under **Layout** and select **Portrait** radio button under **Orientation** heading.
6. Click **Next**.
7. Choose *Soft Gray* style and click **Next**.
8. Type *Category wise Books* in **What title do you want for your report?** text box.
9. Click **Finish** to close the Report Wizard. The Report opens in print preview.



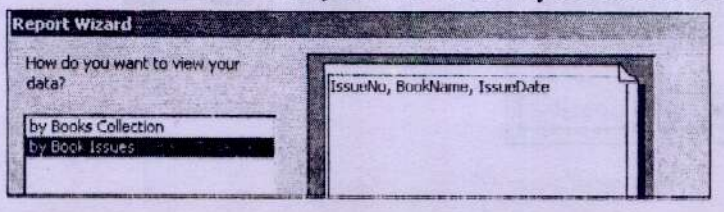

10. Click ☒ button (on the top right corner) to close the report.

## Creating report to display Book issues between desired Period

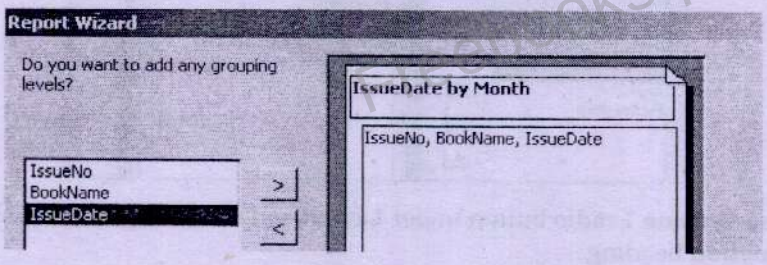
1. Double click **Create report by using wizard** from **Reports** object in database window to open **Report wizard**.
2. Add all the fields from *Books issues b/w desired Period* query and click **Next**.



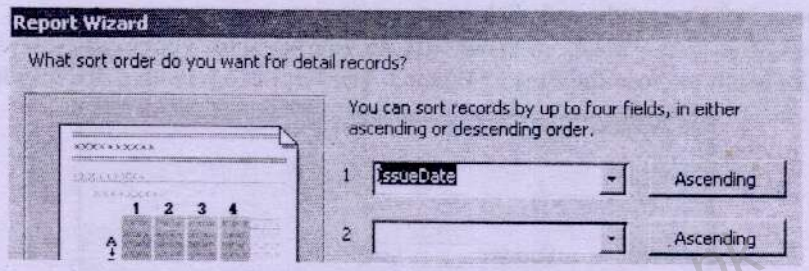

3. Select *by Book Issues* under **How do you want to view your data?** title and click **Next**.



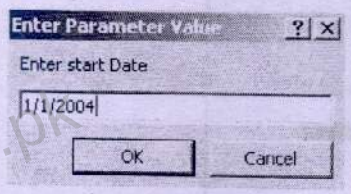
4. Add *IssueDate* in grouping level page and click **Next.**

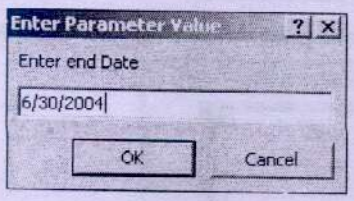


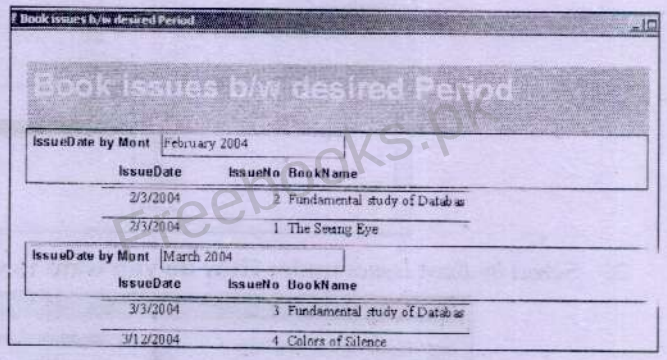

5. Select *IssueDate* field from the drop down list for sorting & click **Next.**




6. Choose **Outline 2** radio button under **Layout** and select **Portrait** radio button under **Orientation** heading and click **Next.**
7. Choose *Soft Gray* style and click **Next.**
8. Type *Book issues b/w desired Periods* in **What title do you want for your report?** text box.
9. Click **Finish** to close the Report Wizard.
10. It is a parameter query so a message will appear asking to enter parameter value.
11. Type *1/1/2004* as first parameter and click **OK.**
12. Type *6/30/2004* as second parameter and press **OK.**
13. Report opens in print preview showing result based on parameter values.
14. Click ✕ button (on the top right corner) to close the report.

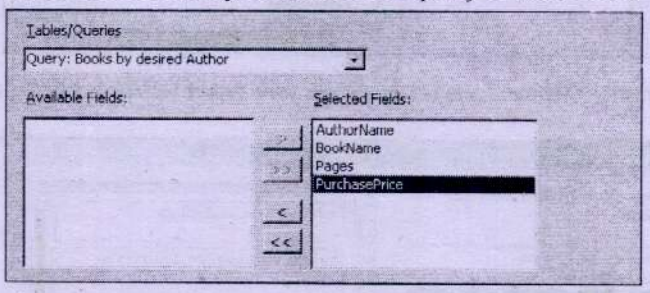


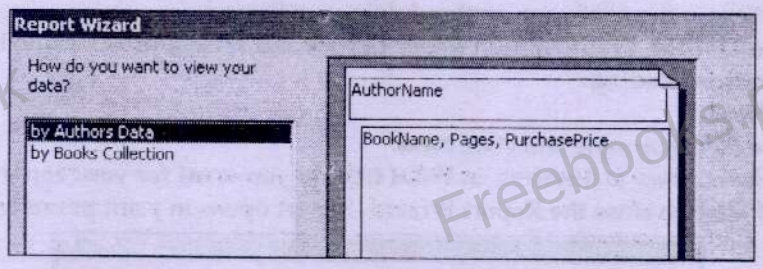



## Creating report to display Books by desired Author

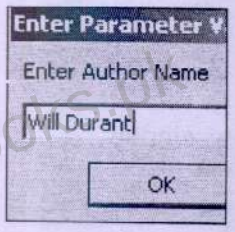
1. Double click **Create report by using wizard** from **Reports** object in database window to open **Report wizard**.
2. Add all the fields from *Books by desired Author* query & click **Next**.

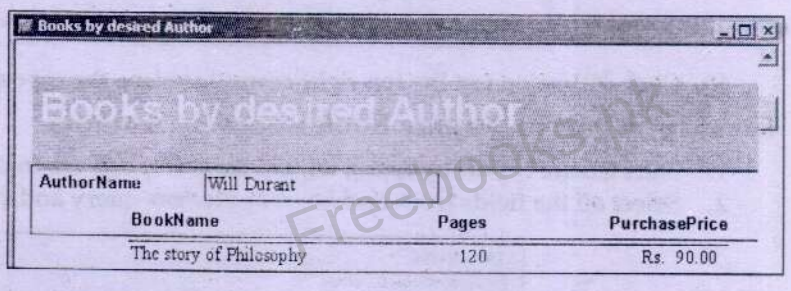



3. Select *by Authors Data* under **How do you want to view your data?** title and click **Next**.




4. Select *BookName* field from the drop down list for sorting and click **Next**.
5. Choose **Align Left 2** radio button under **Layout** and select **Portrait** radio button under **Orientation** heading and press **Next.**
6. Choose *Soft Gray* style and click **Next**.
7. Type *Books by desired Author* in **What title do you want for your report?** text box.
8. Click **Finish** to close the Report Wizard. This report is based on parameter query so before displaying report preview, prompt appear asking to enter parameter value.
9. Type *Will Durant* in the prompt box and click **OK**.
10. Report opens in print preview showing result based on parameter value.

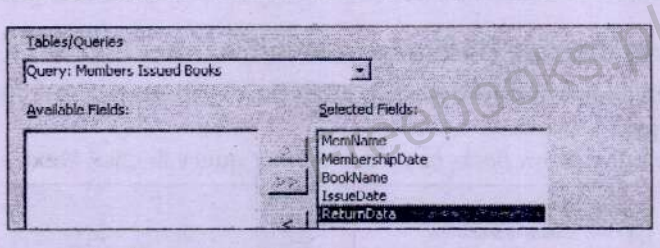


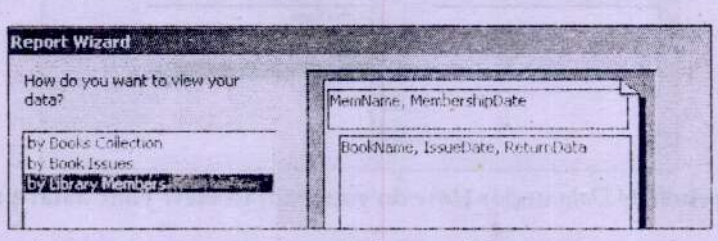

11. Click ✕ button (on the top right corner) to close the report.

## Creating report to display Books issued to Members

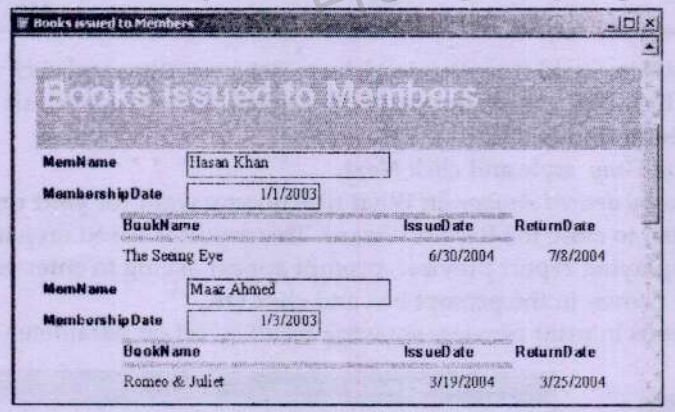
1. Start **Create report by using wizard** from **Reports** object in database window.
2. Select all the fields from *Member issued Books* query and click **Next**.




3. Select *by Library Members* under **How do you want to view your data?** title and click **Next.**



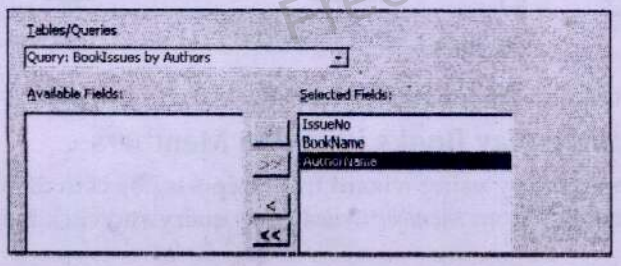

4. Click **Next** in the next page, without doing anything.
5. Choose **Outline 1** radio button under **Layout** and select **Portrait** radio button under **Orientation** heading.
6. Click **Next**.
7. Choose *Soft Gray* style and click **Next**.
8. Type *Books issued to Members* in **What title do you want for your report?** text box.
9. Press **Finish** to close the Report Wizard. Report opens in print preview.



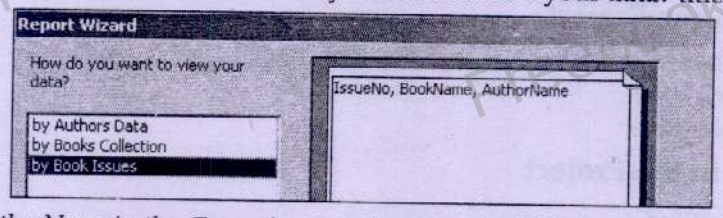

10. Click ✕ button (on the top right corner) to close the report.

## Creating report to display Book issues by Authors

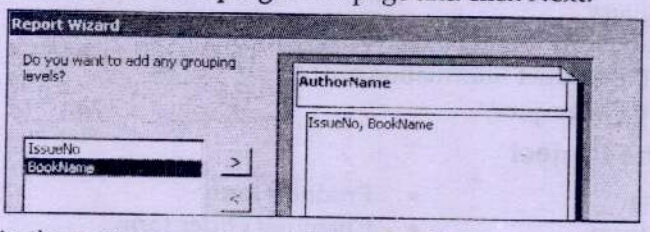
1. Start **Create report by using wizard** from **Reports** tab in database window.
2. Select all the fields from *BookIssues by Authors* query and click **Next.**



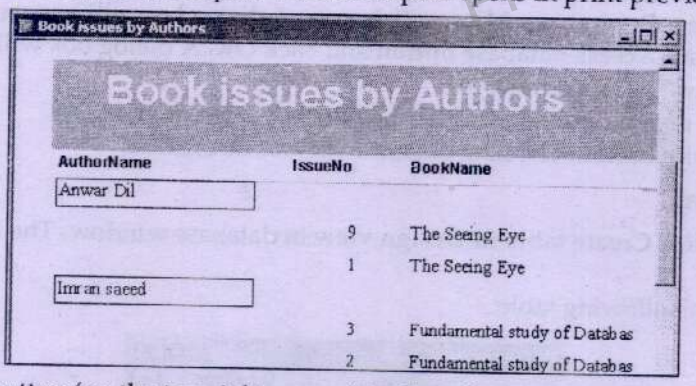
3. Select *by Book Issues* under **How do you want to view your data?** title & press **Next**.




4. Add *AuthorName* in the **Grouping Level** page and click **Next**.




5. Press **Next** in the next page, without doing anything.
6. Choose **Stepped** radio button under **Layout** and select **Portrait** radio button under **Orientation** heading.
7. Click **Next**.
8. Choose *Soft Gray* style and click **Next**.
9. Type *Book issues by Authors* in **What title do you want for your report?** text box.
10. Click **Finish** to close the Report Wizard. Report opens in print preview.




11. Click ☒ button (on the top right corner) to close the report.

## Project 2 Sales Management System

### Project Objectives

The main objectives of the project are as follows:

- Organize information about different entities
- Organize information about customers
- Organize information about sales persons
- Organize information about orders
- Organize information about total sale
- Analyze sales
- Provide easy and attractive interface

## Tables Used in the Project

- Customers table
- Sales Persons table
- Order Details table
- Products table
- Orders table

## Queries Used in the Project

- Products List query
- Customers List query
- Order with Customer Information query
- Order Details with Product Information query
- Sales Analysis by Order query

## Forms Used in the Project

- Customers form
- Sales Persons form
- Products form
- Customer Order form

## Reports Used in the Project

- Orders report
- Products List report
- Sales Analysis by Order report

## Designing the Database

1. Click **Start > Programs > Microsoft Access**. A dialog box will appear.
2. Select **Blank Access database option** and click **OK**. A dialog box will appear.
3. Enter "Sales" in **File name** box.
4. Select the desired location to save the database from **Save in** box.
5. Click **Create** button. The new database file will be created.

## Creating Tables

1. Double click **Create table in Design view** in database window. The Design view will appear.
2. Create the following table:

Customers : Table

| Field Name | Data Type |
|---|---|
| CustomerID | AutoNumber |
| FirstName | Text |
| LastName | Text |
| Address | Text |
| City | Text |
| PhoneNumber | Text |

3. Set **CustomerID** as primary key and save the table as **Customers**.
4. Place the cursor in **CustomerID** field.
5. Select **Long Integer** from **Field Size** property and type **\C0000** in **Format** property at the bottom of design view.
6. Create the following table:

Products : Table

| Field Name | Data Type |
|---|---|
| ProductID | AutoNumber |
| ProductName | Text |
| UnitsInStock | Number |
| UnitsOnOrder | Number |
| UnitPrice | Currency |

7. Set **ProductID** as primary key and save the table as **Products**.
8. Place the cursor in **ProductID** field.
9. Select **Long Integer** from **Field Size** property and type **\P0000** in **Format** property at the bottom of design view.
10. Create the following table:

**Sales Persons : Table**

| Field Name | Data Type |
|---|---|
| SalesPersonID | AutoNumber |
| FirstName | Text |
| LastName | Text |
| WorkPhone | Text |
| HireDate | Date/Time |

11. Set **SalesPersonID** as primary key and save the table as **Sales Persons**.
12. Place the cursor in **SalesPersonID** field.
13. Select **Long Integer** from **Field Size** property and type **\S00** in **Format** property at the bottom of design view.
14. Create the following table:

**Orders : Table**

| Field Name | Data Type |
|---|---|
| OrderID | AutoNumber |
| CustomerID | Number |
| OrderDate | Date/Time |
| SalesPersonID | Number |

15. Set **OrderID** as primary key and save the table as **Orders**.
16. Place the cursor in **OrderID** field.
17. Select **Long Integer** from **Field Size** property and type **\O0000** in **Format** property at the bottom of design view.
18. Create the following table:

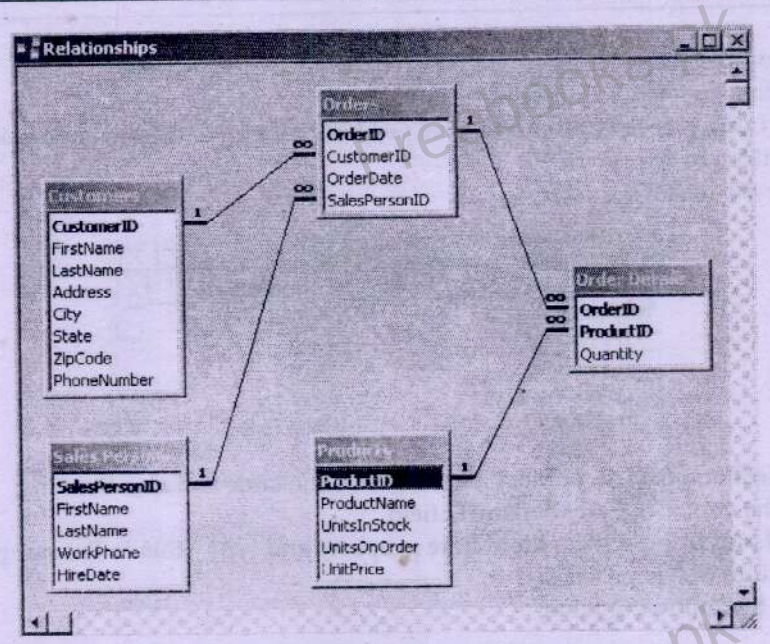
**Order Details : Table**

| Field Name | Data Type |
|---|---|
| OrderID | Number |
| ProductID | Number |
| Quantity | Number |

19. Set **OrderID** and **ProductID** as primary key and save the table as **Orders Details**.

## Creating Relationship

1. Click **Tools > Relationships**. A dialog box will appear.
2. Select all tables one by one from the dialog box and click **Add** button.
3. Click **Close** button. The dialog box will close and **Relationships** window will appear.
4. Create relationship according to the following figure:




## Entering Data

1. Add the following data in **Customers** table.

Customers : Table

| CustomerID | First Name | Last Name | Address | City | Phone Number |
|---|---|---|---|---|---|
| C0001 | Babar | Laeeq | Madina Town | Faisalabad | 1234567 |
| C0002 | Ehsan | Mahmood | Defense | Lahore | 2231251 |
| C0003 | Naveed | Ghauri | Mall Road | Lahore | 5432123 |
| C0004 | Faisal | Akbar | Gulshan Colony | Karachi | 5235123 |
| C0005 | Mahmood | Saleem | Gulgasht Colony | Multan | 6423744 |
| C0006 | Jameel | Ali | Ali Town | Multan | 8286628 |
| C0007 | Ahsan | Raheem | Gulshan e Johar | Karachi | 7373623 |
| C0008 | Shaukat | Shehzad | Officers Colony | Faisalabad | 6737373 |
| C0009 | Adnan | Khalil | Khayber Colony | Peshawer | 8723789 |
| C0010 | Daood | Jaffer | Hayatabad | Peshawer | 6987252 |

2. Add the following data in **Products** table.

Products : Table

| ProductID | Product Name | Units In Stock | Units On Order | UnitPrice |
|---|---|---|---|---|
| P0001 | Celeron® at 2.0GHz | 50 | 0 | 899 |
| P0002 | Pentium® IV at 2.6GHz | 25 | 5 | 1,099 |
| P0003 | Pentium® IV at 3.0GHz | 125 | 15 | 1,399 |
| P0004 | Pentium® III Notebook at 800 MHz | 25 | 50 | 1,599 |
| P0005 | Pentium® IV Notebook at 2.0GHz | 15 | 25 | 2,599 |
| P0006 | 17" CRT Monitor | 50 | 0 | 499 |
| P0007 | 19" CRT Monitor | 25 | 10 | 899 |
| P0008 | 21" CRT Monitor | 50 | 20 | 1,599 |
| P0009 | 2 Years On Site Service | 15 | 20 | 299 |
| P0010 | 4 Years On Site Service | 25 | 15 | 399 |
| P0011 | Multi Media Projector | 10 | 0 | 1,245 |
| P0012 | Digital Camera - 2.0 megapixels | 40 | 0 | 249 |
| P0013 | Digital Camera - 4.0 megapixels | 50 | 15 | 450 |
| P0014 | HD Floppy Disks (50 pack) | 500 | 200 | 10 |
| P0015 | CD-R (25 pack spindle) | 100 | 50 | 15 |
| P0016 | Digital Scanner | 15 | 3 | 180 |
| P0017 | Serial Mouse | 150 | 50 | 70 |
| P0018 | Trackball | 55 | 0 | 60 |
| P0019 | Joystick | 250 | 100 | 40 |
| P0020 | Wireless broadband router | 35 | 10 | 190 |

3. Add the following data in **Sales Persons** table.

Sales Persons : Table

| SalesPersonID | FirstName | LastName | WorkPhone | HireDate |
|---|---|---|---|---|
| S01 | Naeem | Akhter | 3212345 | 2/3/2000 |
| S02 | Mobeen | Ahmed | 2526231 | 2/10/2001 |
| S03 | Shoaib | Bukhari | 2876988 | 3/15/1999 |
| S04 | Kareem | Sheikh | 8745362 | 11/24/2002 |
| S05 | Rasheed | Chaudry | 8760966 | 1/21/2003 |
| (AutoNumber) | | | | |

Record: 1 of 5

4. Add the following data in **Orders** table.

Orders : Table

| OrderID | Customer ID | Order Date | SalesPersonID |
|---|---|---|---|
| O0001 | 0004 | 4/15/2003 | S01 |
| O0002 | C0003 | 4/18/2003 | S02 |
| O0003 | C0006 | 4/18/2003 | S03 |
| O0004 | C0007 | 4/18/2003 | S04 |
| O0006 | C0001 | 4/21/2003 | S05 |
| O0007 | C0002 | 4/21/2003 | S01 |
| O0008 | C0002 | 4/22/2003 | S02 |
| O0009 | C0001 | 4/22/2003 | S03 |
| O0010 | C0002 | 4/22/2003 | S04 |
| (AutoNumber) | | 5/22/2008 | |

Record: 1 of 9

5. Add the following data in **Order Details** table.

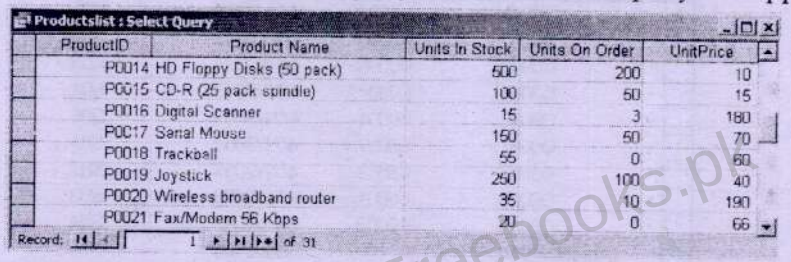
Order Details : Table

| Order ID | Product ID | Quantity |
|---|---|---|
| O0001 | P0013 | 1 |
| O0001 | P0014 | 4 |
| O0001 | P0027 | 1 |
| O0002 | P0001 | 1 |
| O0002 | P0006 | 1 |
| O0002 | P0020 | 1 |
| O0002 | P0022 | 1 |
| O0003 | P0005 | 1 |
| O0003 | P0020 | 1 |
| O0003 | P0022 | 1 |
| O0004 | P0003 | 1 |
| O0004 | P0010 | 1 |
| O0004 | P0022 | 2 |
| O0006 | P0007 | 1 |
| O0006 | P0014 | 10 |
| O0007 | P0028 | 1 |
| O0007 | P0030 | 3 |
| O0008 | P0001 | 1 |
| O0008 | P0004 | 3 |
| O0008 | P0008 | 4 |
| O0008 | P0011 | 2 |

## Creating Queries

1. Click **Queries** in **Database** window.
2. Double click **Create query in design view**. The **Query Design** window will appear containing the **Show Table** dialog box.
3. Select **Products** table and click **Add**.
4. Select the following fields from **Field** list box at the bottom of the window.


5. Save the query as **Products List.**
6. Move to the datasheet view of the query. The result of the query will appear.

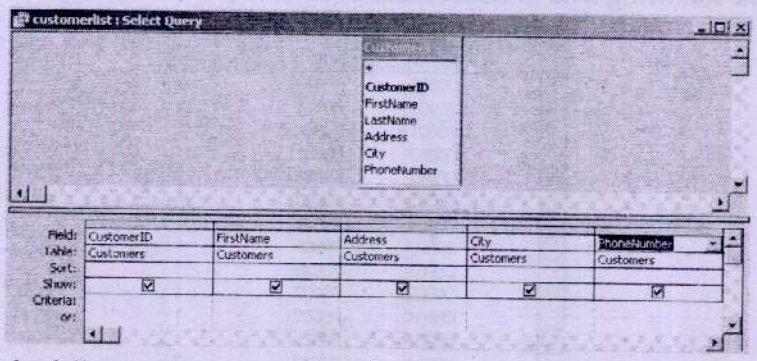


Productslist : Select Query

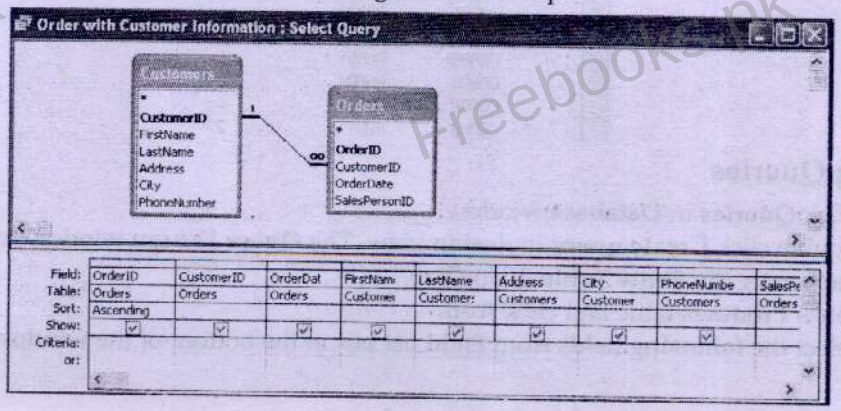
| ProductID | Product Name | Units In Stock | Units On Order | UnitPrice |
|---|---|---|---|---|
| P0014 | HD Floppy Disks (50 pack) | 500 | 200 | 10 |
| P0015 | CD-R (25 pack spindle) | 100 | 50 | 15 |
| P0016 | Digital Scanner | 15 | 3 | 180 |
| P0017 | Serial Mouse | 150 | 50 | 70 |
| P0018 | Trackball | 55 | 0 | 60 |
| P0019 | Joystick | 250 | 100 | 40 |
| P0020 | Wireless broadband router | 35 | 10 | 190 |
| P0021 | Fax/Modem 56 Kbps | 20 | 0 | 66 |

Record: 1 of 31

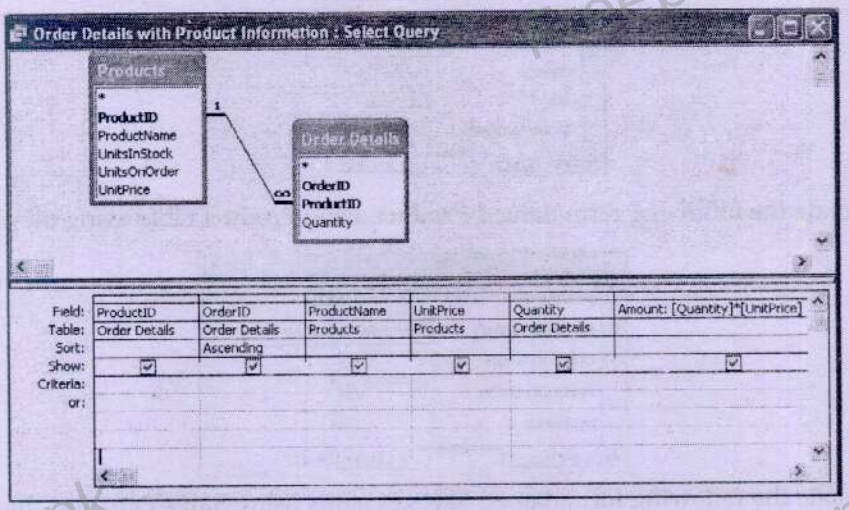
7. Create the following query named **Customers List** using **Customers** table using the above steps.




8. Create the following query named **Orders with Customer Information** using **Customers** and **Orders** tables using the above steps.

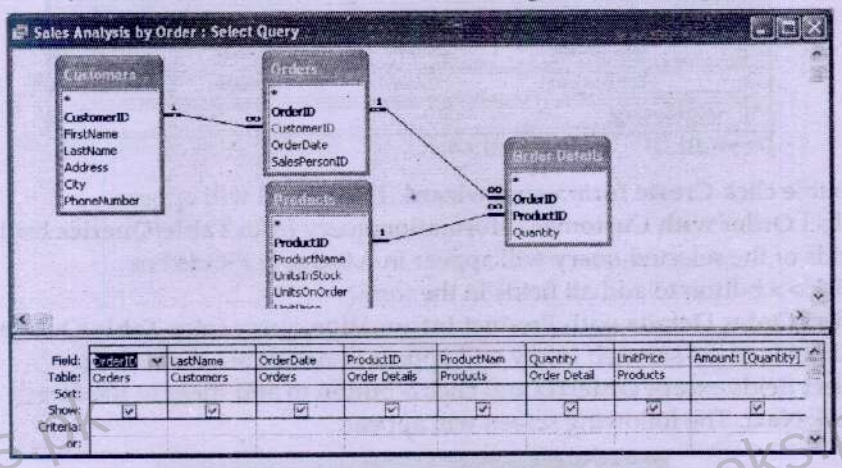


9. Create the following query named **Order Details with Product Information** using **Products** and **Order Details** tables using the above steps.



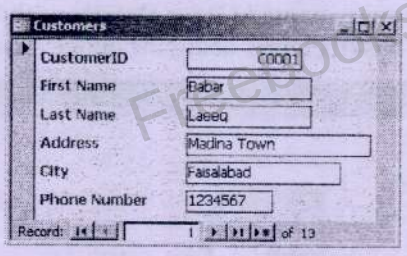

**Note:** The above query uses a calculated field to find the total amount. It uses the formula **Amount: [Quantity] * [UnitPrice]**. The formula contains field names in brackets. The result is calculated by multiplying both fields.

10. Create the following query named **Sales Analysis by Order** using **Customers, Products, Orders** and **Order Details** tables using the above steps.

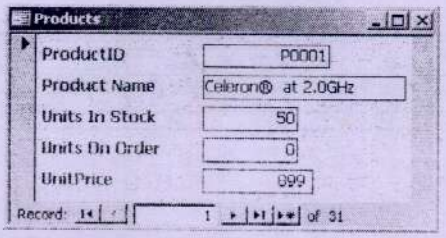



## Creating Forms

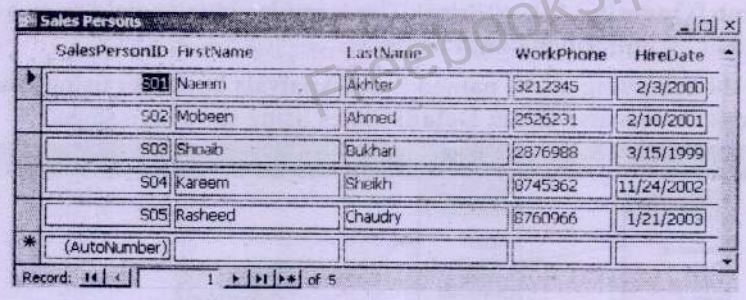
1. Double click **Create form using wizard**. The wizard will appear.
2. Select **Customers** table from **Table/Queries** list box. All fields of the selected table will appear in **Available Fields** box.
3. Click >> button to add all fields in the form.
4. Click **Next.**
5. Select **Columnar** from form type and click **Next.**
6. Select **Sumi Painting** as form style and click **Next.**
7. Type **Customers Form** as form title.
8. Click **Finish**. The following form will appear.



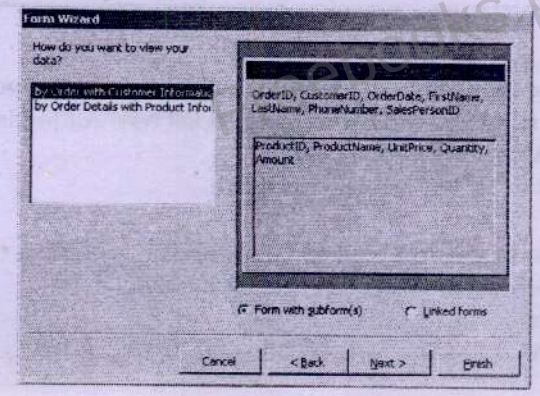

9. Create the following form named **Product** using **Product** table using the above steps.



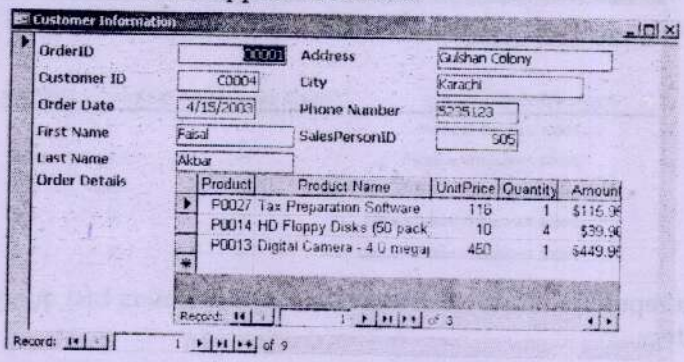

10. Create the following form named **Sales Persons** using **Sales Persons** table using the above steps.




11. Double click **Create form using wizard**. The wizard will appear.
12. Select **Order with Customer Information** query from **Table/Queries** list box. All fields of the selected query will appear in **Available Fields** box.
13. Click >> button to add all fields in the form.
14. Select **Order Details with Product Information** query from **Table/Queries** list box. All fields of the selected query will appear in **Available Fields** box.
15. Select fields except **OrderID** and click > button to add them in the form.
16. Click **Next**. The following screen will appear.

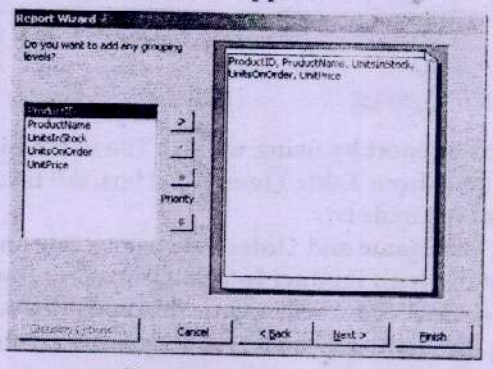


17. Click **Next**.
18. Select **Datasheet** layout.
19. Select **Sumi Painting** as form style and click **Next**.
20. Type **Customers Order** as title for main form and **Order Detail** as title for sub form.
21. Click Finish. The form will appear as follows:

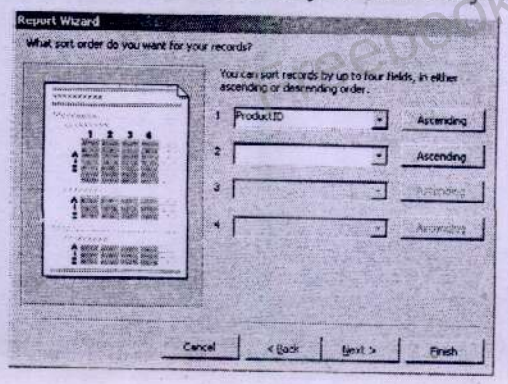



## Creating Reports

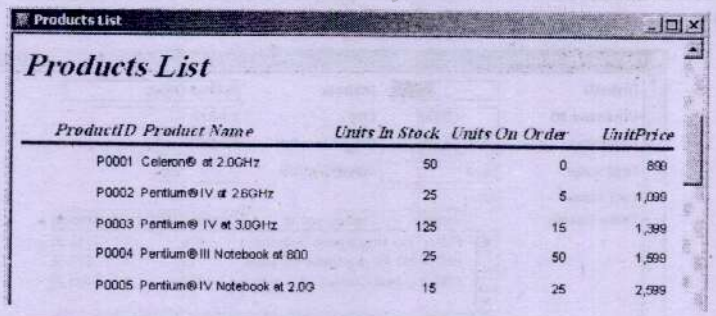
1. Click **Reports** button in database window.
2. Double click **Create report by using wizard**. The wizard will appear.
3. Select **Products** table from **Table/Queries** list box. All fields of the selected table will appear in **Available Fields** box.
4. Click >> button to add all fields in the report.
5. Click **Next.** The following screen will appear.




6. Click **Next**. A new screen will appear.
7. Select **ProductID** from **What sort order do you want for your records?** option.



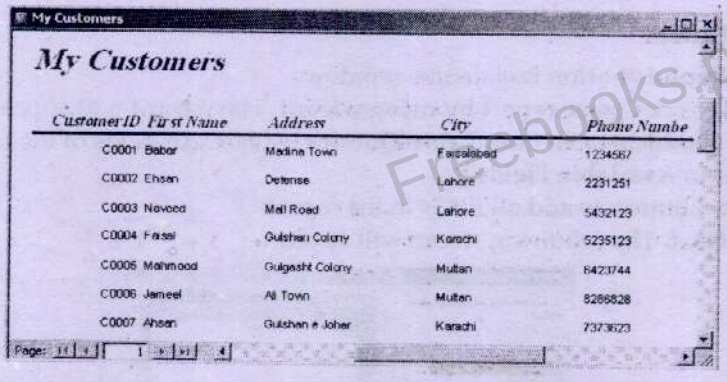
8. Select the desired **Layout** and **Orientation** and click **Next**.
9. Select the report style and click **Next**.
10. Type **Product List** as report name.
11. Click **Finish**. The report will appear.



Products List

| ProductID | Product Name | Units In Stock | Units On Order | UnitPrice |
|---|---|---|---|---|
| P0001 | Celeron® at 2.0GHz | 50 | 0 | 800 |
| P0002 | Pentium® IV at 2.6GHz | 25 | 5 | 1,099 |
| P0003 | Pentium® IV at 3.0GHz | 125 | 15 | 1,399 |
| P0004 | Pentium® III Notebook at 800 | 25 | 50 | 1,599 |
| P0005 | Pentium® IV Notebook at 2.0G | 15 | 25 | 2,599 |

12. Create a report names **My Customers** using **Customers List** query by using the above steps.

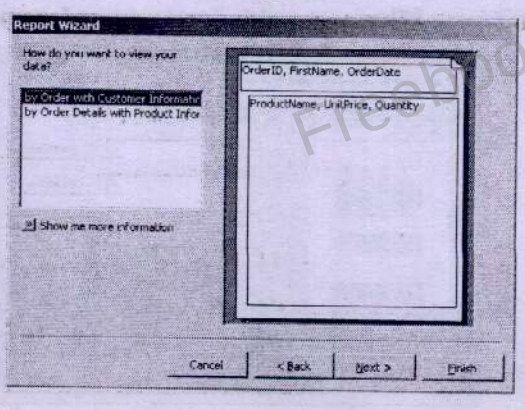


My Customers

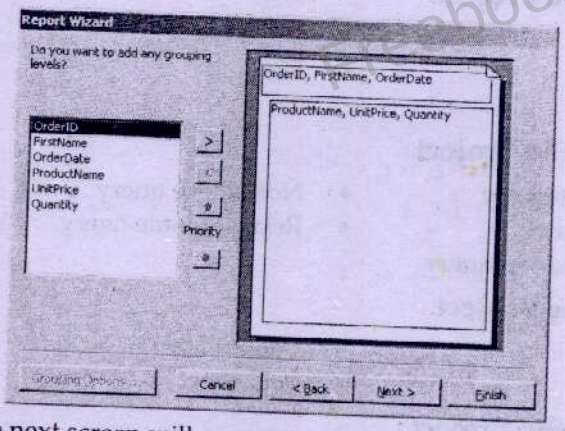
| CustomerID | First Name | Address | City | Phone Numbe |
|---|---|---|---|---|
| C0001 | Babar | Madina Town | Faisalabad | 1234567 |
| C0002 | Ehsan | Defense | Lahore | 2231251 |
| C0003 | Noveed | Mall Road | Lahore | 5432123 |
| C0004 | Faisal | Gulshan Colony | Karachi | 5235123 |
| C0005 | Mahmood | Gulgasht Colony | Multan | 8423744 |
| C0006 | Jameel | Ali Town | Multan | 8286828 |
| C0007 | Ahsan | Gulshan e Johar | Karachi | 7373623 |

Page: 1

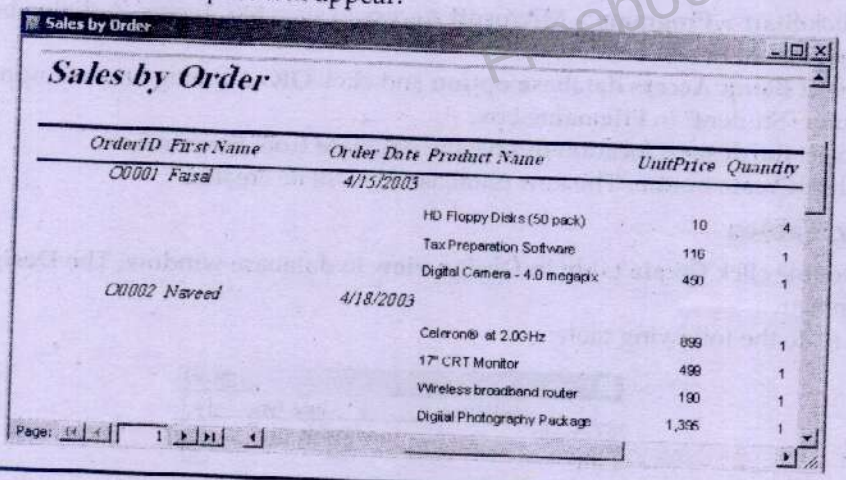
13. Double click **Create report by using wizard**. The wizard will appear.
14. Select **Products** table from **Table/Queries** list box. All fields of the selected table will appear in **Available Fields** box.
15. Select **OrderID, FirstName** and **OrderDate** one by one and click > button.
16. Select **Order Details with Product Information** query from **Table/Queries** list box. All fields of the selected table will appear in **Available Fields** box.
17. Select **UnitPrice, ProductName** and **Quantity** fields one by one and click > button.
18. Click **Next**. The following screen will appear.



19. Click **Next**. The following screen will appear.




20. Click **Next**. The next screen will appear.
21. Click **Next** button.
22. Select the desired **Layout** and **Orientation** and click **Next**.
23. Select the report style and click **Next**.
24. Type **Sales Analysis by Order** as report name.
25. Click **Finish**. The report will appear.



Sales by Order

| OrderID | First Name | Order Date | Product Name | UnitPrice | Quantity |
|---|---|---|---|---|---|
| O0001 | Faisal | 4/15/2003 | | | |
| | | | HD Floppy Disks (50 pack) | 10 | 4 |
| | | | Tax Preparation Software | 116 | 1 |
| | | | Digital Camera - 4.0 megapix | 450 | 1 |
| O0002 | Naveed | 4/18/2003 | | | |
| | | | Celeron® at 2.0GHz | 899 | 1 |
| | | | 17" CRT Monitor | 499 | 1 |
| | | | Wireless broadband router | 190 | 1 |
| | | | Digital Photography Package | 1,395 | 1 |

Page: 1

## Project 3 Student Management System

### Project Objectives

The main objectives of the project are as follows:

- Organize the records of the students in electronic database
- Provide facilities to process data of the students
- Provide the facility to search the required data
- Provide the facility to store data about different subjects
- Provide the facility to store data about different classes
- Provide the facility to store data about results
- Provide the facility to store data about fees

## Tables Used in the Project

- Class table
- Student table
- Result table
- Subject table
- Fee table

## Queries Used in the Project

- Students Result query
- StudentFSD query
- Student Information query
- NotFullFee query
- ResultAGrade query

## Forms Used in the Project

- Class form
- Result form
- Subject form
- Fee form

## Reports Used in the Project

- Student report
- NotFullFee report
- ResultFSD report
- Subject report
- ResultAGrade report

## Designing the Database

1. Click **Start > Programs > Microsoft Access** to start MS Access. A dialog box will appear.
2. Select **Blank Access database option** and click **OK**. A dialog box will appear.
3. Enter "Student" in **File name** box.
4. Select the desired location to save the database from **Save in** box.
5. Click **Create** button. The new database file will be created.

## Creating Tables

1. Double click **Create table in Design view** in database window. The **Design view** will appear.
2. Create the following table:

Class : Table

| Field Name | Data Type |
|---|---|
| ClassID | Number |
| Name | Text |

3. Set **ClassID** as primary key and save the table as **Class**.
4. Create the following table:

Subject : Table

| Field Name | Data Type |
|---|---|
| SubjectID | Number |
| Name | Text |

5. Set **SubjectID** as primary key and save the table as **Subject**.
6. Create the following table:

Student : Table

| Field Name | Data Type |
|---|---|
| RollNo | Number |
| ClassID | Number |
| Name | Text |
| FName | Text |
| BirthDate | Date/Time |
| Address | Text |
| City | Text |
| Phone | Text |

7. Set **RollNo** as primary key and save the table as **Student**.
8. Create the following table:

**Result : Table**

| Field Name | Data Type |
|---|---|
| RollNo | Number |
| SubjectID | Number |
| TotalMarks | Number |
| ObtMarks | Number |

9. Set **RollNo** and **SubjectID** as primary key and save the table as **Result**.
10. Create the following table:

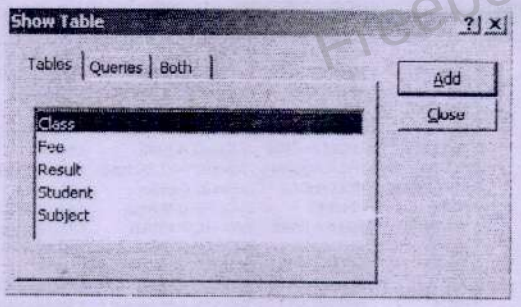
**Fee : Table**

| Field Name | Data Type |
|---|---|
| ReceiptNo | Number |
| DepositDate | Date/Time |
| RollNo | Number |
| Amount | Currency |

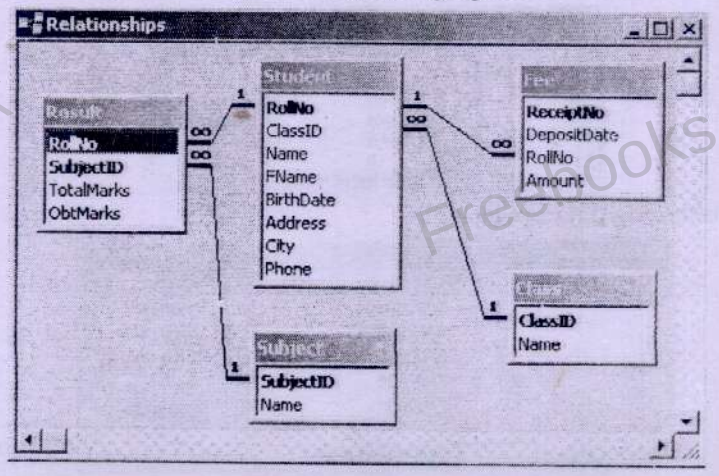
11. Set **ReceiptNo** as primary key and save the table as **Fee**.

## Creating Relationship

1. Click **Tools > Relationships**. A dialog box will appear.




2. Select all tables one by one from the dialog box and click **Add** button.
3. Click **Close** button. The dialog box will close and **Relationships** window will appear.
4. Create relationship according to the following figure:



## Entering Data

1. Add the following data in **Class** table.

Class : Table

| ClassID | Name |
|---|---|
| 1 | D.COM-I |
| 2 | D.COM-II |
| 3 | B.COM-I |
| 4 | B.COM-II |
| 5 | I.COM-I |
| 6 | I.COM-II |

Record: 1

2. Add the following data in **Subject** table.

Subject : Table

| SubjectID | Name |
|---|---|
| 3401 | English |
| 3402 | Business Stat |
| 3403 | Business IT |
| 3404 | Accounting |
| 3405 | Economics |
| 3406 | Urdu |
| 3407 | Islamiat |
| 3408 | Pak Studies |
| 3409 | Physics |
| 3410 | Chemistry |
| 0 | |

Record: 1 of 10

3. Add the following data in **Student** table.

Student : Table

| RollNo | ClassID | Name | FName | BirthDate | Address | City | Phone |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Asghar | A.Z.K Shargil | 17-Oct-02 | P-100 | FAISALABAD | 7888234 |
| 2 | 1 | FURQAN | MUHAMMADAL | 20-Oct-02 | H-1010/P | FAISALABAD | 7575100 |
| 3 | 1 | JAZ | MUHAMMADSI | 20-Oct-02 | ST#45 G.M AB | MIANWALI | 784200 |
| 4 | 1 | MAHMISH | HAJIKARAMDIN | 20-Oct-02 | Block #5 H.N | LAHORE | 678768 |
| 5 | 1 | NIDA | SHEIKHALI | 22-Oct-02 | G-98 St#2 | FAISALABAD | 768584 |
| 101 | 2 | MUBEEN | ABDULCHEEM | 22-Oct-02 | H-109/B | MULTAN | 565564 |
| 102 | 2 | NOSHARWAN | FAZALHUSAIN | 22-Oct-02 | P-77 Block#3 | FAISALABAD | 78456 |
| 103 | 2 | MUHAMMAD | MULLAHDAD | 22-Oct-02 | P-876/A | MULTAN | 565676 |
| 104 | 2 | REDA | HAMEED | 23-Oct-02 | G.M.Abad | FAISALABAD | 443372 |
| 105 | 2 | WAJAHAT | MALIKANWAR | 18-Oct-02 | P-1010/B | FAISALABAD | 856698 |
| 201 | 3 | MIRZA | ALIAHMAD | 20-Oct-02 | H-190/B | GUJRANWALA | 785643 |
| 202 | 3 | FAKIR | ABDULJABBAF | 20-Oct-02 | H-1009/B Mirza | CHINIOT | 786090 |
| 203 | 3 | ASIYA | ALIAHMAD | 22-Oct-02 | P-109/B | HYDERABAD | 674443 |
| 204 | 3 | RAJA | HAYYATALI | 22-Oct-02 | Y-98 Block#T | FAISALABAD | 67565 |
| 205 | 3 | Q | GULZARALI | 22-Oct-02 | P-1010/T | PATTOKI | 565666 |
| 0 | 0 | | | | | | |

Record: 1 of 15

4. Add the following data in **Result** table.

Result : Table

| RollNo | SubjectID | TotalMarks | ObtMarks |
|---|---|---|---|
| 1 | 3401 | 100 | 91 |
| 1 | 3402 | 100 | 88 |
| 1 | 3403 | 100 | 75 |
| 1 | 3404 | 100 | 65 |
| 1 | 3405 | 100 | 69 |
| 0 | 0 | 0 | 0 |

Record: 5 of 5

5. Add the following data in **Fee** table.

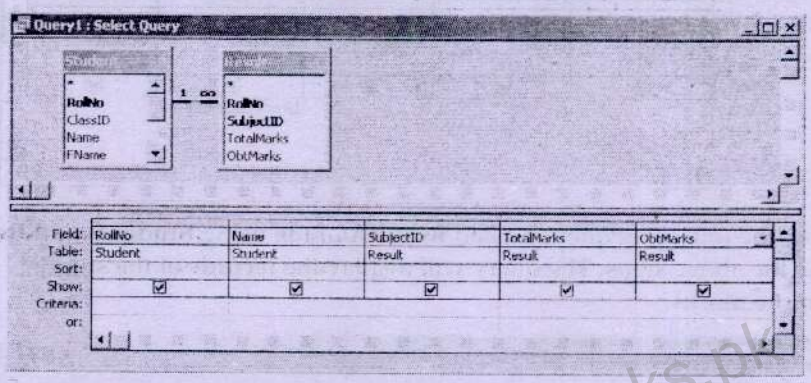
Fee : Table

| ReceiptNo | DepositDate | RollNo | Amount |
|---|---|---|---|
| 1 | 1/4/2008 | 1 | Rs.10,000 |
| 2 | 4/4/2008 | 2 | Rs.0,000 |
| 3 | 7/4/2008 | 5 | Rs.10,000 |
| 4 | 7/4/2008 | 3 | Rs.5,000 |
| 5 | 7/4/2008 | 4 | Rs.10,000 |
| 0 | | 0 | Rs.0 |

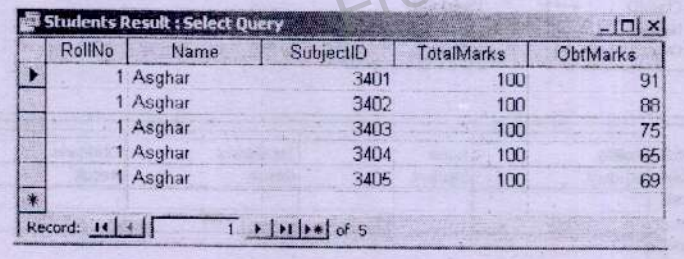
Record: 1 of 5

## Creating Queries

1. Click **Queries** in **Database** window.
2. Double click **Create query in design view**. The **Query Design** window will appear containing the **Show Table** dialog box.
3. Select **Student** table and click **Add**.
4. Select **Result** table and click **Add**.
5. Select the following fields from **Field** list box at the bottom of the window.




11. Save the query as **Students Result**.
12. Move to the datasheet view of the query. The result of the query will appear.

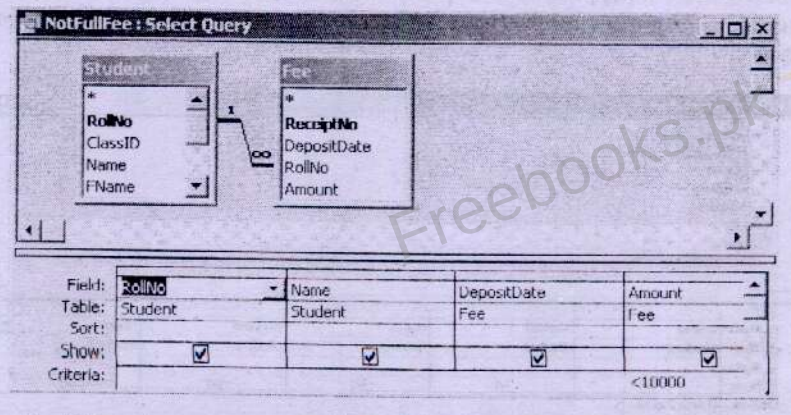


Students Result : Select Query

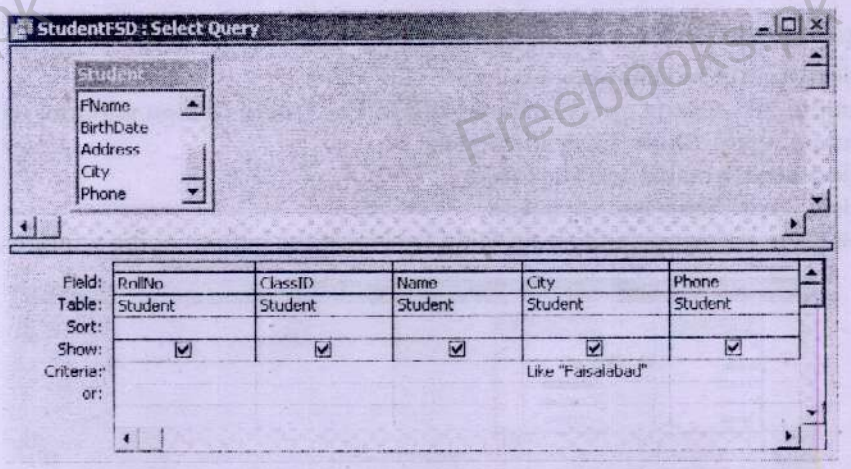
| RollNo | Name | SubjectID | TotalMarks | ObtMarks |
|---|---|---|---|---|
| 1 | Asghar | 3401 | 100 | 91 |
| 1 | Asghar | 3402 | 100 | 88 |
| 1 | Asghar | 3403 | 100 | 75 |
| 1 | Asghar | 3404 | 100 | 65 |
| 1 | Asghar | 3405 | 100 | 69 |

Record: 1 of 5

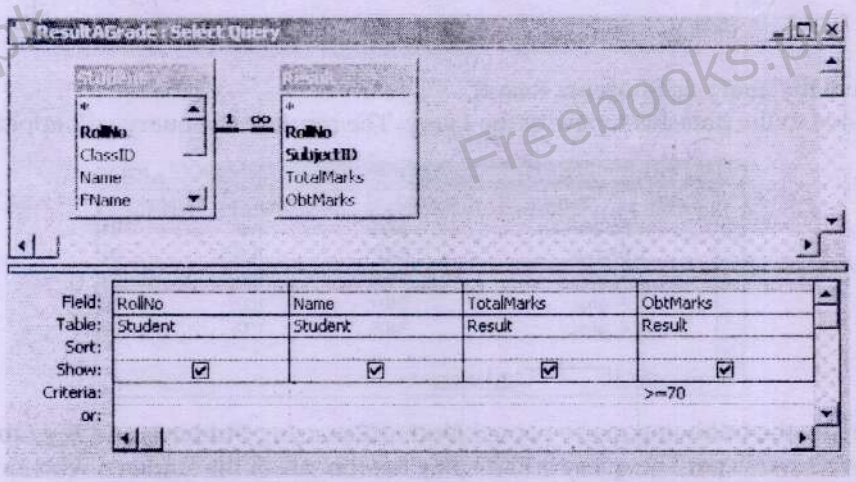
13. Create the following query named **NotFullFee** using **Students** and **Fee** tables using the above steps. The query will display the records of the students who have not paid full fee.



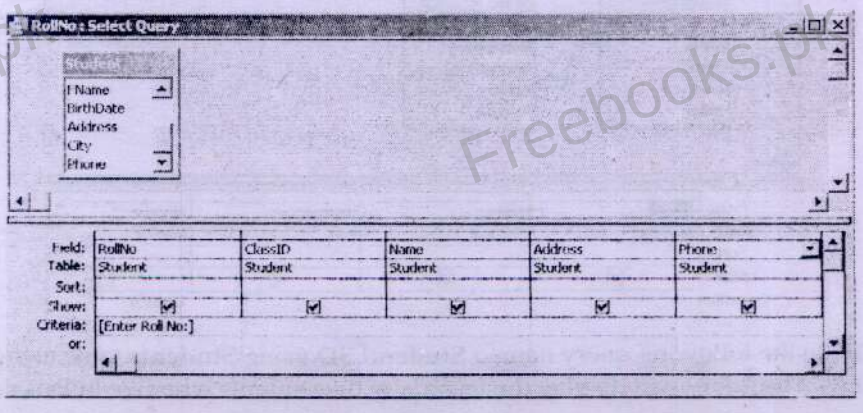

14. Create the following query named **StudentFSD** using **Students** table using the above steps. The query will display the records of the students who live in Faisalabad.




15. Create the following query named **ResultAGrade** using **Students** and **Result** tables using the above steps. The query will display the records of the students who got 70 ore more marks.

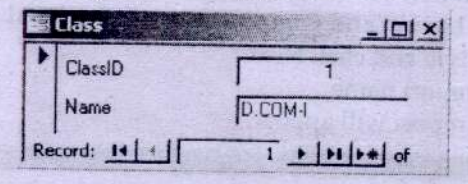



16. Create the following parametric query named **RollNo** using **Students** table using the above steps. The query will input Roll No as parameter and display the record of that Roll No.

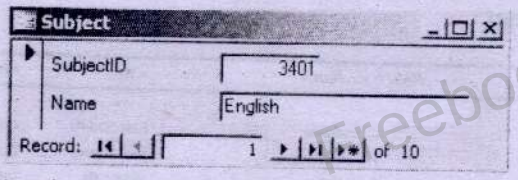


## Creating Forms

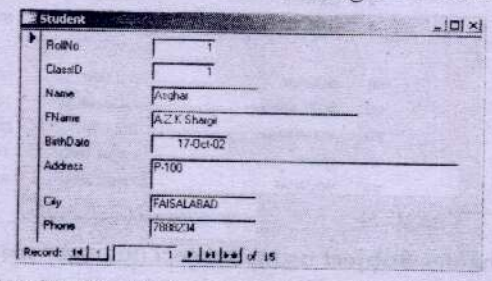
1. Double click **Create form using wizard**. The wizard will appear.
2. Select **Class** table from **Table/Queries** list box. All fields of the selected table will appear in **Available Fields** box.
3. Click >> button to add all fields in the form.
4. Click **Next**.
5. Select **Columnar** from form type and click **Next**.
6. Select **Standard** as form style and click **Next**.
7. Type **Class** as form title.
8. Click **Finish**. The following form will appear.



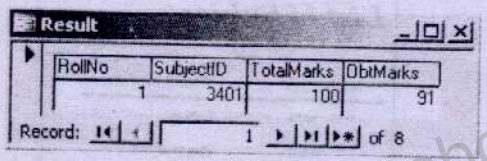

9. Create the following form named **Subject** using **Subject** table using the above steps.



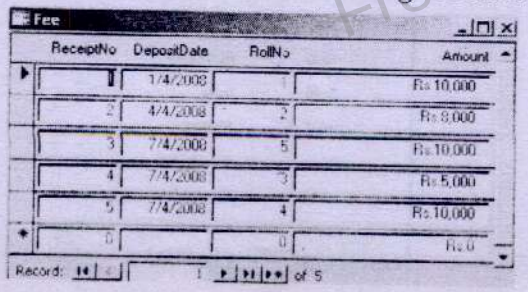

10. Create the following form named **Student** using **Student** table using the above steps.




11. Create the following justified form named **Result** using **Result** table using the above steps.

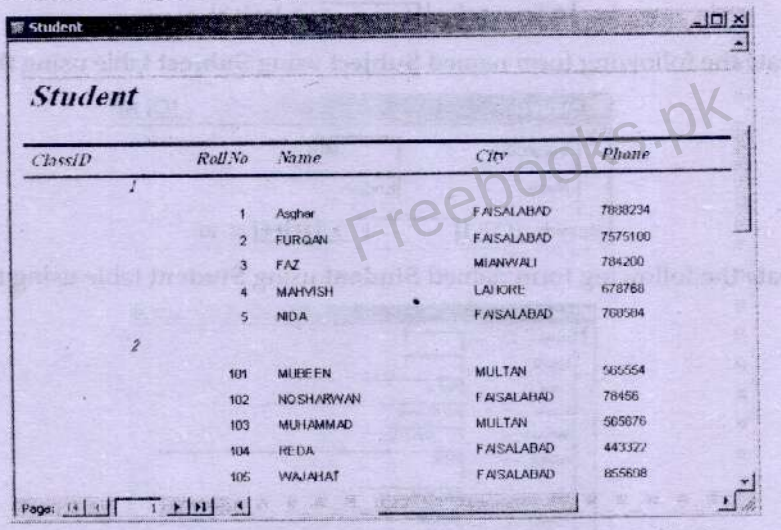



12. Create the following tabular form named **Fee** using **Fee** table using the above steps.



## Creating Reports

1. Click **Reports** button in database window.
2. Double click **Create report by using wizard**. The wizard will appear.
3. Select **Student** table from **Table/Queries** list box. All fields of the selected table will appear in **Available Fields** box.
4. Click >> button to add all fields in the report.
5. Click **Next**. The following screen will appear.
6. Select **ClassID** for grouping.
7. Click **Next** button.
8. Select RollNo from first box to sort the records in the report.
9. Select the desired **Layout** and **Orientation** and click **Next**.
10. Select the report style and click **Next**.
11. Type **Student** as report name.
12. Click **Finish**. The report will appear.



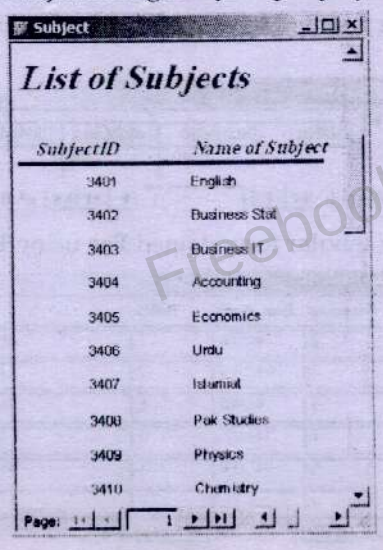
Student

*Student*

| ClassID | RollNo | Name | City | Phone |
|---|---|---|---|---|
| 1 | | | | |
| | 1 | Asgher | FAISALABAD | 7888234 |
| | 2 | FURQAN | FAISALABAD | 7575100 |
| | 3 | FAZ | MIANWALI | 784200 |
| | 4 | MAHVISH | LAHORE | 678768 |
| | 5 | NIDA | FAISALABAD | 760504 |
| 2 | | | | |
| | 101 | MUBEEN | MULTAN | 565554 |
| | 102 | NOSHARWAN | FAISALABAD | 78456 |
| | 103 | MUHAMMAD | MULTAN | 565676 |
| | 104 | REDA | FAISALABAD | 443322 |
| | 105 | WAJAHAT | FAISALABAD | 855608 |

Page: 1

13. Create a report names **Subject** using **Subject** query by using the above steps.



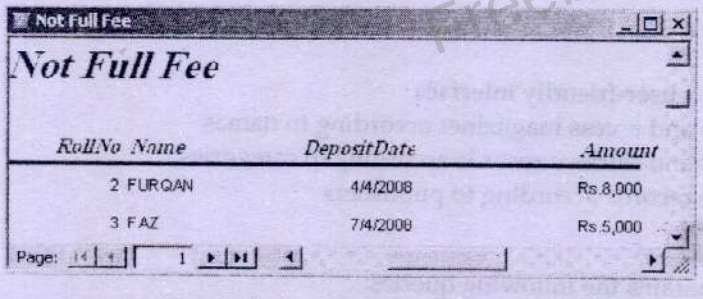
Subject

*List of Subjects*

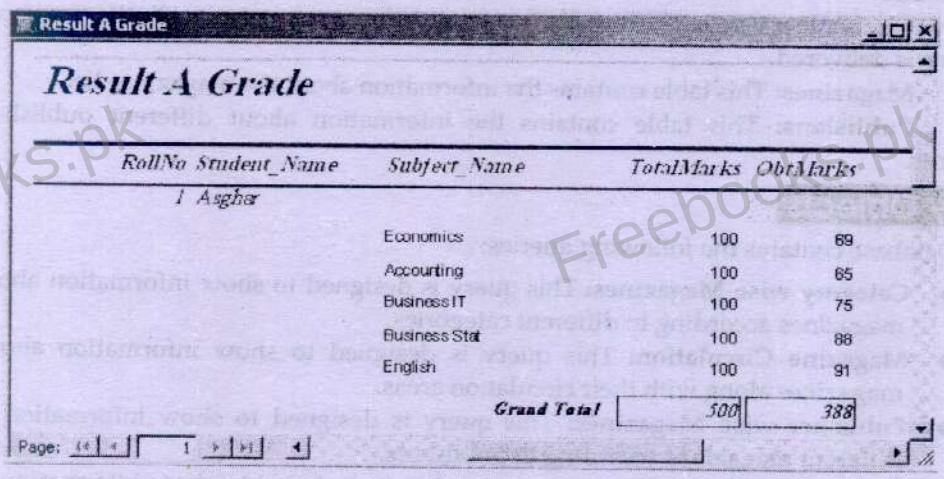
| SubjectID | Name of Subject |
|---|---|
| 3401 | English |
| 3402 | Business Stat |
| 3403 | Business IT |
| 3404 | Accounting |
| 3405 | Economics |
| 3406 | Urdu |
| 3407 | Islamiat |
| 3408 | Pak Studies |
| 3409 | Physics |
| 3410 | Chemistry |

Page: 1

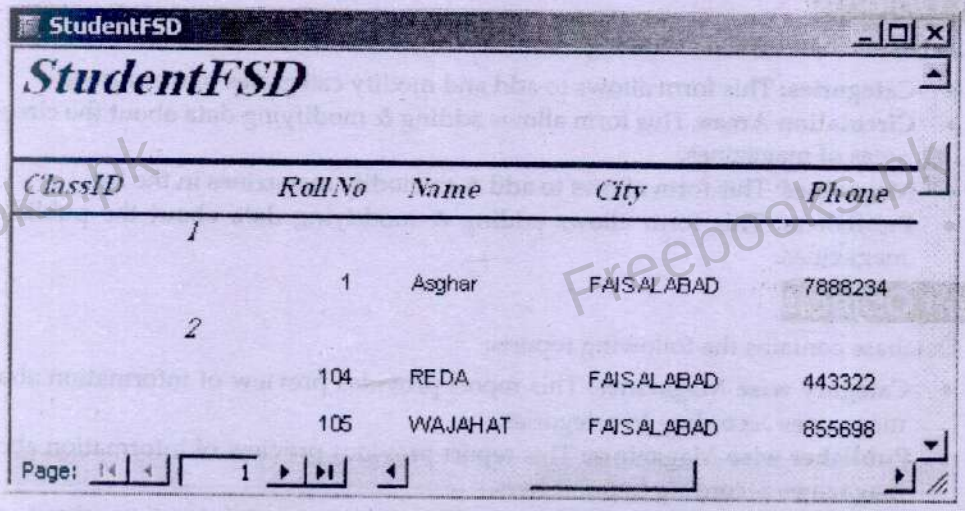
14. Create a report names **Not Full Fee** using **NotFullFee** query by using the above steps.




15. Create a report names **Result A Grade** using **Students, Subject** and **Result tables** by using the above steps.




16. Create a report names **ResultFSD** using **ResultFSD** query by using the above steps.



## Project 4 | Magazine Database

### Goals

- Provide a user-friendly interface
- Manage and access magazines according to names
- Manage and retrieve records according to categories
- Manage records according to publishers

### Table Objects

Database contains the following queries:

- **Categories:** This table contains the specific information about different categories of magazines.
- **Circulation Areas:** This table contains information about all the areas where magazine is delivered.
- **Magazines:** This table contains the information about the magazines list.
- **Publishers:** This table contains the information about different publishers of magazines.

### Query Objects:

Database contains the following queries:

- **Category wise Magazines:** This query is designed to show information about the magazines according to different categories.
- **Magazine Circulation:** This query is designed to show information about the magazines along with their circulation areas.
- **Publisher wise Magazines:** This query is designed to show information about different magazines according to publishers.
- **Desired Magazine:** This parameter query is designed to show information about that particular magazine which is entered as parameter.

### Form Objects

Database contains the following forms:

- **Categories:** This form allows to add and modify categories of magazines.
- **Circulation Areas:** This form allows adding & modifying data about the circulation areas of magazines.
- **Magazines:** This form allows to add & to modify magazines in the list.
- **Publishers:** This form allows adding & modifying data about the publishers of magazines.

### Report Objects

Database contains the following reports:

- **Category wise Magazines:** This report provides preview of information about the magazines according to categories.
- **Publisher wise Magazines:** This report provides preview of information about the magazines according to publishers.
- **Magazine List:** This report provides preview of information about all the magazines in the list.

## Tasks

Perform the following tasks for developing staff database in Ms Access.

1. Open **Ms Access** and create a new Database named **Magazine db**.
2. Create first table named **Magazines** with the following fields:

**Magazines : Table**

| Field Name | Data Type | Description |
|---|---|---|
| MagId | AutoNumber | |
| MagName | Text | |
| FirstPublished | Number | Year in which magazine was published first time |
| Editor | Text | |
| Assistant Editor | Text | |
| Designer | Text | Person who designs the Magazine title, etc |
| TotalPages | Number | |
| Rate | Currency | Daily/Monthly Price |
| CategoryId | Number | Category to which paper belongs |
| PubId | Number | Printing press of magazine |

3. Create second table named **Categories** with the following fields:

**Categories : Table**

| Field Name | Data Type |
|---|---|
| CategoryId | AutoNumber |
| Categoryname | Text |

4. Create third table named **Circulation Areas** with the following fields:

**Circulation Areas : Table**

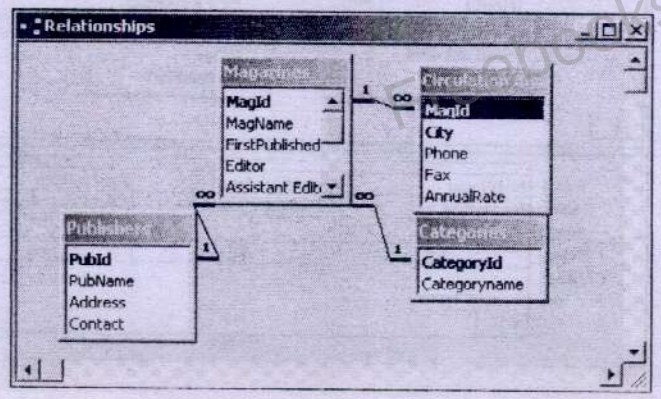
| Field Name | Data Type | Description |
|---|---|---|
| MagId | Number | |
| City | Text | Cities to which magazine is delivered |
| Phone | Text | |
| Fax | Text | |
| AnnualRate | Currency | Rate for subscription of 12 months |

5. Create fourth table named **Publishers** with the following fields:

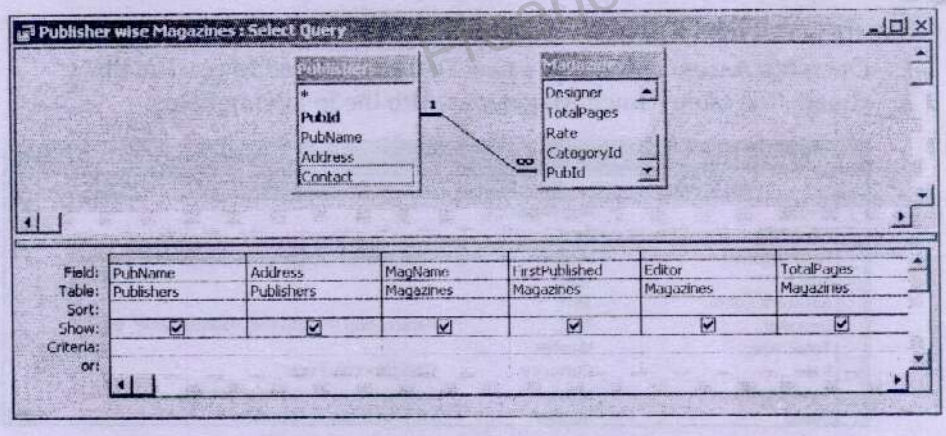
**Publishers : Table**

| Field Name | Data Type |
|---|---|
| PubId | AutoNumber |
| PubName | Text |
| Address | Text |
| Contact | Number |

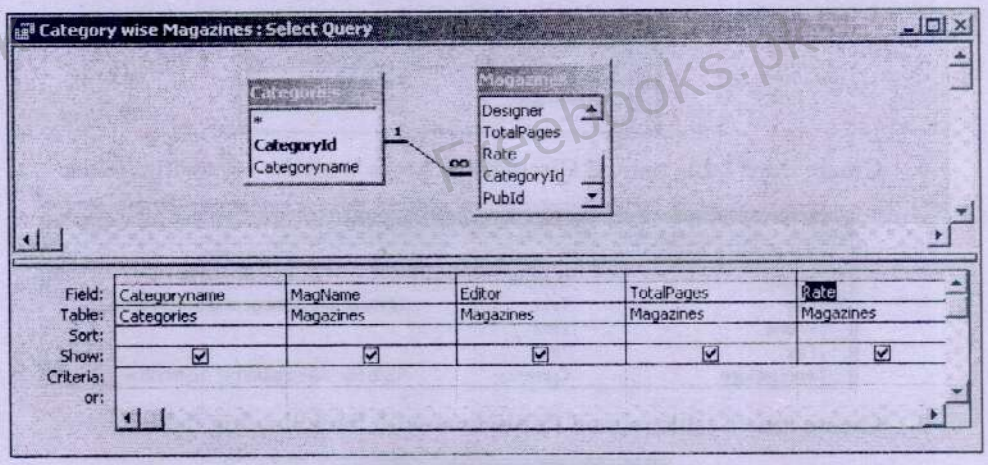
6. Develop **Relationship** among the tables as follows:



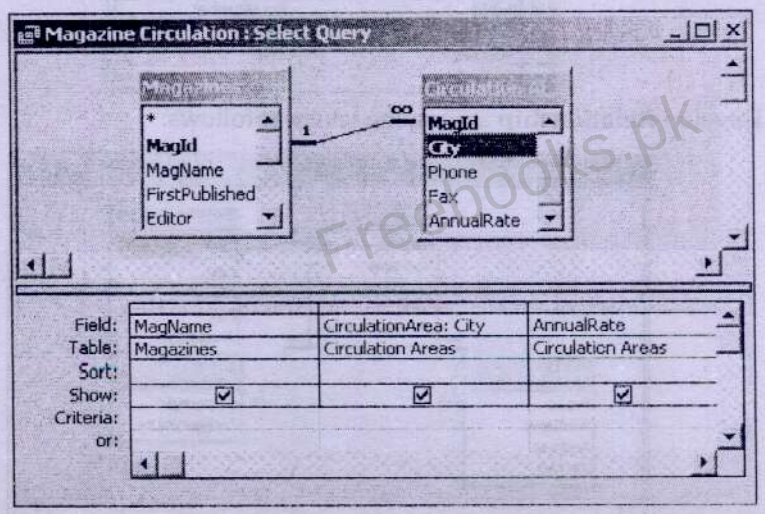
7. Add records in the tables.
8. Design **Publisher wise Magazines** query as follows:



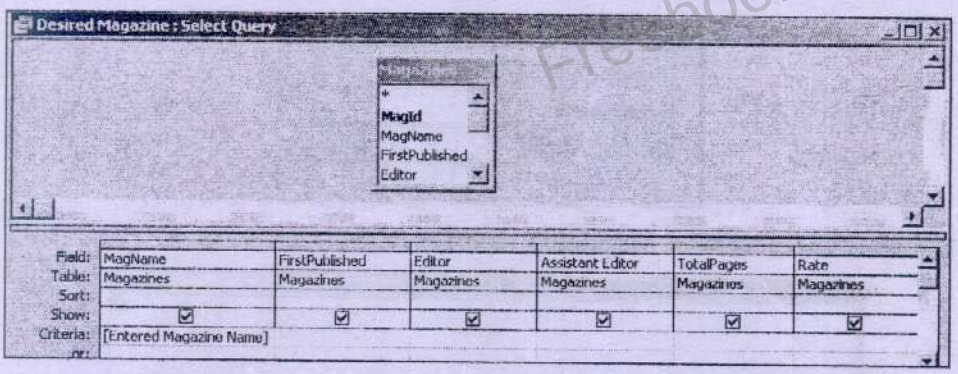

9. Design **Category wise Magazines** query as follows:



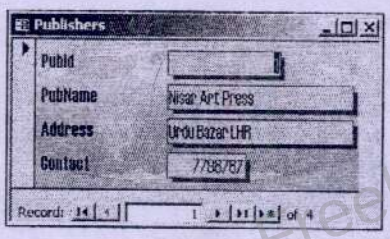

10. Design **Magazines Circulation** query as follows:



11. Design **Desired Magazines** parameter query as follows:

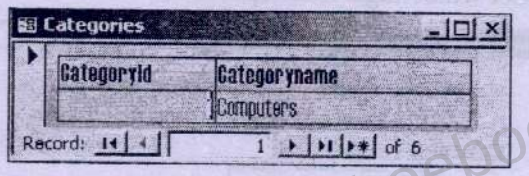



12. Design **Publishers** form as follows:

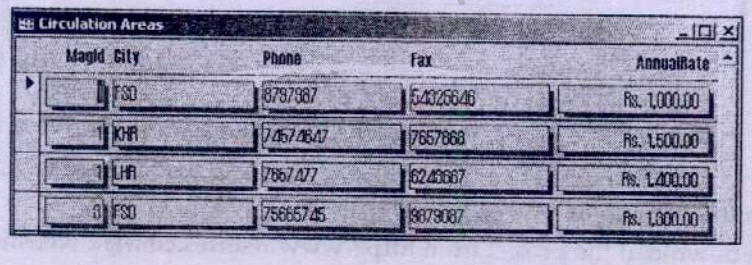



13. Design **Magazines** form as follows:


14. Design **Categories** form as follows:

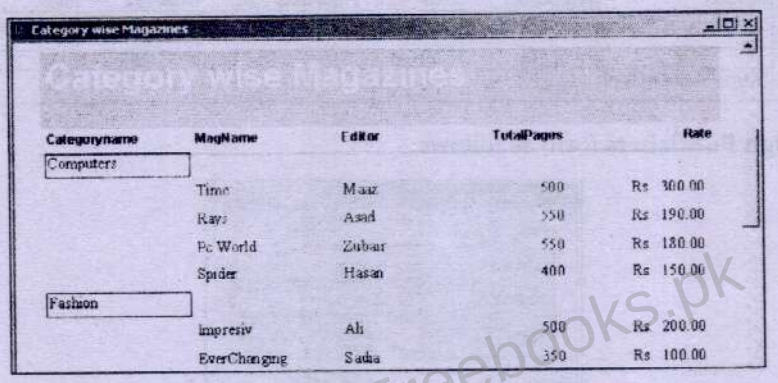



15. Design **Circulation Areas** form as follows:

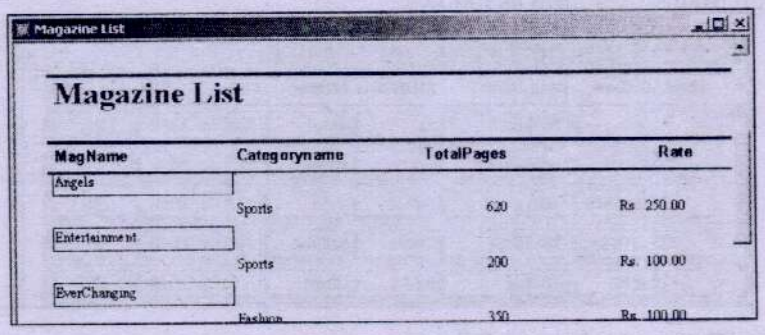


16. Design **Magazine Circulation** form as follows:

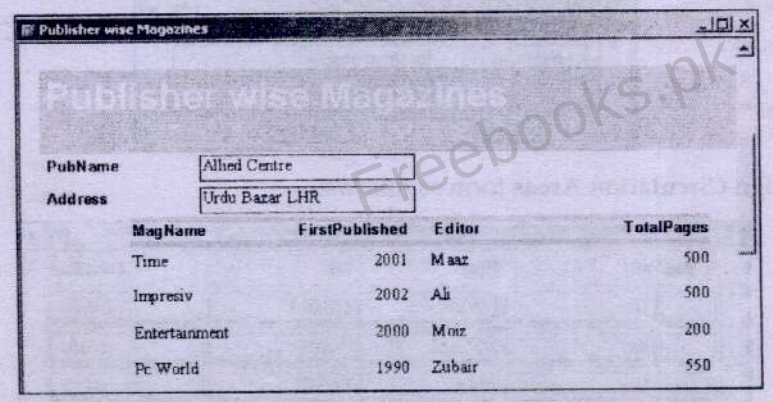

17. Create **Category wise Magazines** report as follows:




18. Create **Magazines List** report as follows:




19. Create **Publisher wise Magazines** report as follows:




20. **Save** the changes and **close** the database.

**Note:** You can download the project from **http://www.itseries.com.pk** website.

## Project 5 Construction Company Database

### Goals

- Provide a user-friendly interface
- Manage & access employee records
- Manage order details of customers
- Mange customer records

### Table Objects:

- **Customers:** This table contains the specific information about each customer.
- **Employees:** This table contains information about the staff working in the company.
- **Orders:** This table contains the information about orders of company.
- **Attendance:** This table contains the employee attendances and work at different orders.

### Query Objects

- **Customer Orders:** This query is designed to show information about the orders placed by different customers.
- **Employee Orders:** This query is designed to show order processing by different employees
- **Orders of entered city:** This parameter query is designed to show orders placed by customers of any particular entered city.

### Form Objects

- **Customers:** This form allows adding & modifying information about each customer.
- **Employees:** This form allows adding & modifying information about the staff working in the company.
- **Orders:** This form allows adding & modifying information about orders of company.
- **Attendance:** This form allows adding & modifying employee attendances and their work at different orders.
- **Customers with orders subform:** This form contains an orders form in it which lets to add and modify customer's data as well as their orders.

### Report Objects

- **Customers:** This report provides preview of specific information about each customer.
- **Employees:** This report provides preview of information about the staff working in the company.
- **Orders:** This report provides preview of orders given to company.
- **Customer Orders:** This report provides preview of information about the customers and orders placed by them.
- **Orders of entered city:** This report is based on a parameter query and generates orders report based on the value of entered parameter.

## Tasks

Perform the following tasks for developing staff database in Ms Access.

1. Open **Ms Access** and create a new Database named **Construction Company**.
2. Create first table named **Employees** with the following fields:

**Employees : Table**

| Field Name | Data Type |
|---|---|
| EmpId | Text |
| EmpName | Text |
| DateHired | Date/Time |
| ContactNumber | Number |
| EmailAddress | Hyperlink |

3. Create second table named **Customers** with the following fields:

**Customers : Table**

| Field Name | Data Type |
|---|---|
| CustomerId | Text |
| CustomerName | Text |
| BillingAddress | Text |
| City | Text |
| ContactNumber | Text |

4. Create third table named **Orders** with the following fields:

**Orders : Table**

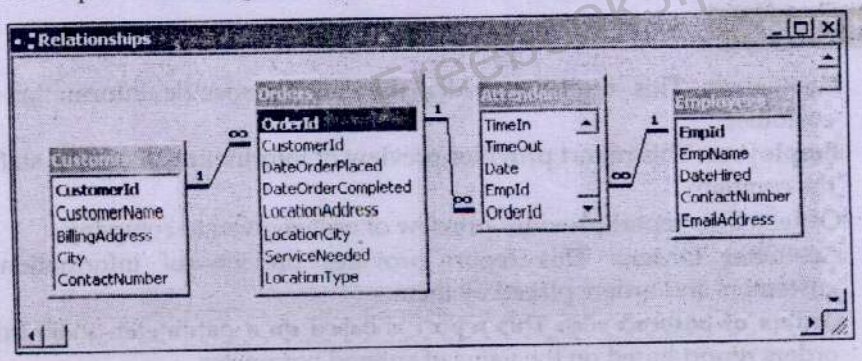
| Field Name | Data Type |
|---|---|
| OrderId | Text |
| CustomerId | Text |
| DateOrderPlaced | Date/Time |
| DateOrderCompleted | Date/Time |
| LocationAddress | Text |
| LocationCity | Text |
| ServiceNeeded | Text |
| LocationType | Text |

5. Create fourth table named **Attendance** with the following fields:

**Attendance : Table**

| Field Name | Data Type |
|---|---|
| AttendanceId | Text |
| TimeIn | Date/Time |
| TimeOut | Date/Time |
| Date | Date/Time |
| EmpId | Text |
| OrderId | Text |

6. Develop **Relationship** among the tables as follows:



7. Add records in the tables.
8. Create **Customer Orders** query as follows:

Customer Orders : Select Query

| CustomerName | OrderId | DateOrderPlaced | DateOrderCompleted | ServiceNeeded | LocationType |
|---|---|---|---|---|---|
| Ahmed | JO00004 | 4/15/2004 | | Concrete poring | Residential |
| Hamza | JO00005 | 2/4/2004 | 2/24/2004 | Escivation | Commercial |
| Talha | JO00006 | 4/19/2004 | | Concrete Wall, 4 ft deep | Residential |
| Hasan | JO00002 | 2/1/2002 | 2/8/2002 | Swimming pool | Commercial |
| Ali | JO00001 | 2/3/2002 | 3/3/2002 | Escivating | Residential |
| Ali | JO00003 | 3/1/2004 | 3/1/2004 | Concrete wall | Commercial |

Record: 1 of 6

9. Create **Employee Orders** query as follows:

Employee Orders : Select Query

| EmpName | AttendanceId | TimeIn | TimeOut | Date | OrderId |
|---|---|---|---|---|---|
| Hasan Ali | DH0003 | 9:00 AM | 4:00 PM | 2/6/2002 | JO00001 |
| Hasan Ali | DH0004 | 9:00 AM | 5:00 PM | 2/7/2002 | JO00001 |
| Hasan Ali | DH0005 | 8:00 AM | 7:00 PM | 2/10/2002 | JO00001 |
| Hasan Ali | CD0002 | 12:00 PM | 4:00 PM | 3/5/2004 | JO00003 |
| Hasan Ali | CD00015 | 1:00 PM | 5:00 PM | 2/11/2004 | JO00005 |
| Hasan Ali | CD00016 | 12:00 PM | 6:00 PM | 2/24/2004 | JO00005 |
| Zubair Malik | DH0006 | 10:00 AM | 7:00 PM | 2/14/2002 | JO00001 |
| Zubair Malik | CD00010 | 9:00 AM | 7:00 PM | 4/16/2004 | JO00004 |
| Zubair Malik | CD00011 | 7:00 AM | 3:00 PM | 4/17/2004 | JO00004 |

10. Design **Orders of entered city** parameter query as follows:

Orders of enetred city : Select Query

| Field: | CustomerName | DateOrderPlaced | DateOrderCompleted | LocationAddress | LocationCity | ServiceNeeded |
|---|---|---|---|---|---|---|
| Table: | Customers | Orders | Orders | Orders | Orders | Orders |
| Sort: | | | | | | |
| Show: | ☑ | ☑ | ☑ | ☑ | ☑ | ☑ |
| Criteria: | | | | | [Enter City] | |

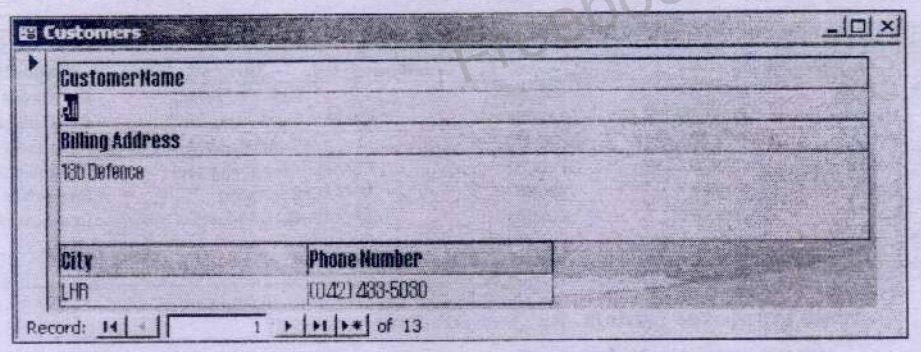
11. Design **Employees** form.

Employees

EmpId: 3001
EmpName: Zaid Tariq
DateHired: 12/8/2000
ContactNumber: 78787878
EmailAddress: zaid@shahnawaz.com

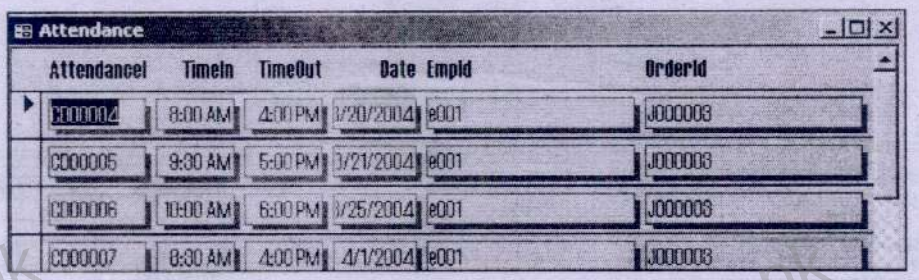
Record: 1 of 5

12. Design **Customers** form as follows:

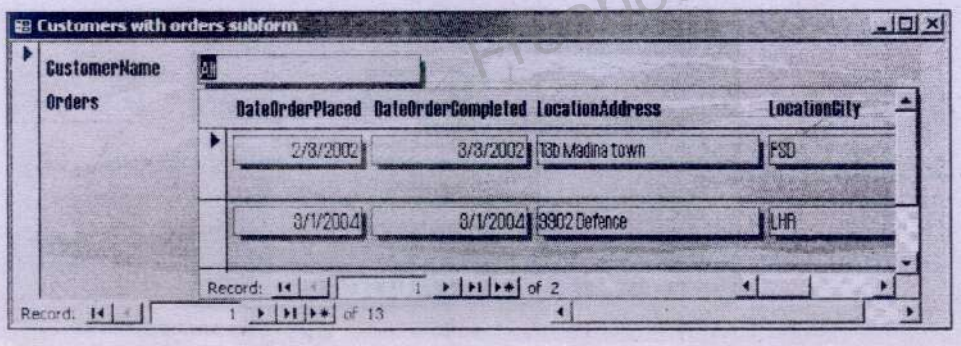



13. Design **Orders** form as follows:


14. Design **Attendance** form as follows:




15. Design **Customers with orders** from as follows:



16. Create **Employees** report as follows:

Employees

*Employees*

| *EmpName* | *EmpId* | *DateHired* | *ContactNumber* | *EmailAddress* |
|---|---|---|---|---|
| Hamza Khan | e005 | 12/30/2001 | 32432424 | khanhamza@shahnawaz.com |
| Hasan Ali | e003 | 6/13/2001 | 64364664 | ali_hasan@shahnawaz.com |
| Talha Ahmed | e004 | 6/13/2001 | 68768768 | talha@shahnawaz.com |
| Zaid Tariq | e001 | 12/13/2000 | 78787878 | zaid@shahnawaz.com |
| Zubair Malik | e002 | 12/30/2000 | 43254555 | zubair_malik@shahnawaz.com |

Page: 1

17. Create **Customers** report as follows:

Customers

*Customers*

| *CustomerName* | *Billing Address* | *City* | *Phone Number* |
|---|---|---|---|
| Ahmed | 12b amtex road | LHR | (042) 434-5680 |
| Ali | 13b Defence | LHR | (042) 433-5030 |
| Ammad | 78c poeples colony | FSD | (041) 685-4332 |
| Anas | 130 shah town | LHR | (042) 856-3251 |

18. Create **Orders** report as follows:

Orders

*Orders*

| *CustomerId* | *OrderId* | *Placed* | *Completed* | *LocationA* | *LocationCi* | *ServiceNeeded* | *LocationT* |
|---|---|---|---|---|---|---|---|
| C0001 | | | | | | | |
| | JO00 | 3/1/2004 | 3/1/2004 | 9902 Defence | LHR | Concrete wall | Commercial |
| | JO00 | 2/3/2002 | 3/3/2002 | 13b Madina to | FSD | Escavating | Residential |
| C0002 | | | | | | | |
| | JO00 | 2/1/2002 | 2/8/2002 | 119 Commerci | LHR | Swimming pool | Commercial |
| C0003 | | | | | | | |
| | JO00 | 4/15/2004 | | 284 Fateh Cen | FSD | Concrete poring | Residential |
| C0004 | | | | | | | |
| | JO00 | 2/4/2004 | 2/24/2004 | 1234 Satteite t | FSD | Escavation | Commercial |
| C0005 | | | | | | | |
| | JO00 | 4/19/2004 | | 20 Music Ave | FSD | Concrete Wall, 4 ft deep | Residential |

Page: 1

19. Create **Customer Orders** report as follows:

Customer Orders

# Customer Orders

| CustomerName | OrderI | DateOrderPlaced | DateOrderCompleted | ServiceNeeded |
|---|---|---|---|---|
| Ahmed | | | | |
| | JO000 | 4/15/2004 | | Concrete poring |
| Ali | | | | |
| | JO000 | 3/1/2004 | 3/1/2004 | Concrete wall |
| | JO000 | 2/3/2002 | 3/3/2002 | Escivating |
| Hamza | | | | |
| | JO000 | 2/4/2004 | 2/24/2004 | Escivation |
| Hasan | | | | |
| | JO000 | 2/1/2002 | 2/8/2002 | Swimming pool |
| Talha | | | | |
| | JO000 | 4/19/2004 | | Concrete Wall, 4 ft deep |

Page: 1

20. Create **Entered City orders** parameter report based on **Orders of entered City** query. When the report is run, a prompt appears asking for the city name. Depending upon that entered city report is generated. For instance if entered city is FSD then report looks like as follows:

Entered City orders

# Entered City orders

| CustomerName | LocationCity | rderPlaced | rCompleted | LocationAddr | ServiceNeeded |
|---|---|---|---|---|---|
| Ali | | | | | |
| | FSD | 2/3/2002 | 3/3/2002 | 13b Madina town | Escivating |
| Talha | | | | | |
| | FSD | 4/19/2004 | | 20 Music Ave | Concrete Wall, 4 ft deep |

Page: 1

21. **Save** the changes and **close** the database.

**Note:** You can download the project from **http://www.itseries.com.pk** website

# Excellent CDs of IT Series


**IT Series** has launched the training CDs for the people who want to learn computer skills. Every person can uses these CDs to learn computer skills easily and cost-effectively.

## Complete Training CD of MS Word & Internet

The training CD has the following salient features:

- Learn Microsoft Word in a few days
- Ideal for beginners
- Easy to follow videos with visual clarity
- Cost effective learning
- Learn at your own pace
- All lectures are in Urdu language so any person to learn computer skills easily
- Complete coverage of all topics including hand-on practices
- All CDs contain menu-based commands to learn different topics
- User-friendly interface
- Simple and easy language
- Easy installation and registration procedures for users
- Complete technical and educational support available for registered users

The following CDs are coming soon:

- Complete Training CD of Programming in C Language
- Complete Training CD of Microsoft Access & PowerPoint
- Complete Training CD of Web Designing
- and more...

For further inquiries, please contact:

**URL:** www.itseries.com.pk
**Email:** info@itseries.com.pk
**Mob:** 0321 661 56 56

# Excellent Books of IT Series


URL: http//www.itseries.com.pk
Email: info@itseries.com.pk
Mobile: 0321 661 56 56

Also Available at: Kitab Markaz, Amin Pur Bazar, Faisalabad.
Imtiaz Book Depot, Urdu Bazar, Lahore.

ISBN 978 - 969 - 9157 - 04- 2
9789699157042