2

Data Storage, Retrieval and Data Base Management Systems

LEARNING OBJECTIVES

♦ Understand number systems and file organizations.
♦ Understand the concepts of Database Management System, their types and components of Databases
♦ Understand the difference between various structures of database systems.
♦ Understand the need of data warehousing and data mining in business applications.

2.0 Introduction to Number Systems

We use the decimal numbers or the decimal number system for our day-to-day activities. As we all know, in the decimal number system there are ten digits – 0 through 9. But computers understand only 0s and 1s - the machine language. But using 0s and 1s to program a computer is a thing in the past. Now we can use the decimal numbers, the alphabets and special characters like +,-,*,?/, etc. for programming the computer. Inside the computer, these decimal numbers, alphabets and the special characters are converted into 0s and 1s, so that the computer can understand what we are instructing it to do. To understand the working of a computer, the knowledge of binary, octal and hexadecimal number systems is essential.

2.1 Decimal Number System

The base or radix of a number system is defined as the number of digits it uses to represent the numbers in the system. Since decimal number system uses 10 digits - 0 through 9 - its base or radix is 10. The decimal number system is also called base-10 number system. The weight of each digit of a decimal number system depends on its relative position within the number. For example, consider the number 3256.

3256 = 3000 + 200 + 50 + 6, or, in other words,
3256 = 3 x 10^3 + 2 x 10^2 + 5 x 10^1 + 6 x 10^0

From the above example, we can see that the weight of the nth digit of the number from the right hand side is equal to \( n^{th} \) digit \( \times 10^{n-1} \) which is again equal to \( nth \ digit \times (base)^{n-1} \). The
number system, in which the weight of each digit depends on its relative position within the numbers, is called the *positional number system*.

### 2.2 Binary Number System

The base or radix of the binary number system is 2. It uses only two digits - 0 and 1. Data is represented in a computer system by either the presence or absence of electronic or magnetic signals in its circuitry or the media it uses. This is called a binary or two-state representation of data since the computer is indicating only two possible states or conditions. Media such as magnetic disks and tapes indicate these two states by having magnetized spots whose magnetic fields can have two different directions or polarities. These binary characteristics of computer circuitry and media are the primary reasons why the binary number system is the basis for data representation in computers. Thus, for electronic circuits, the conduction state (ON) represents a ONE and the non-conducting state (OFF) represents a ZERO.

Therefore, as mentioned earlier, the binary number system has only two symbols, 0 and 1. The binary symbol 0 or 1 is commonly called a bit, which is a contraction of the term binary digit. In the binary system, all numbers are expressed as groups of binary digits (bits), that is, as groups of 0s and 1s. Just as in any other number system, the value of a binary number depends on the position or place of each digit in a grouping of binary digits. The values are based on the right to left position of digits in a binary number, using the power of 2 as position values. For example, consider the binary number 10100.

\[
10100 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \\
= 16 + 0 + 4 + 0 + 0 \\
= 20
\]

Table 1 gives the binary equivalents of the decimal numbers from 0 to 20.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>15</td>
<td>1111</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>16</td>
<td>10000</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>17</td>
<td>10001</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>18</td>
<td>10010</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>19</td>
<td>10011</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>20</td>
<td>10100</td>
</tr>
<tr>
<td>10</td>
<td>1010</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Binary-decimal Conversion: To convert a binary number to its decimal equivalent, we use the following expression:

The weight of the n\textsuperscript{th} bit of a number from the right hand side = n\textsuperscript{th} bit \times 2^{n-1}

After calculating the weight of each bit, they are added to get the decimal value for real numbers as shown in the following examples:

101 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 4 + 0 + 1 = 5

1010 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 8 + 0 + 2 + 0 = 10

1111 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 4 + 2 + 1 = 15

For fractional numbers as shown in the following example:

1.001 = 1 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 1 + 0 + 0 + .125 = 1. 125

Decimal-binary Conversion: Decimal numbers are converted into binary by a method called Double Dabble Method. In this method, the mantissa part of the number is repeatedly divided by 2 with storing the reminders, which will be either 0 or 1. This division is continued till the mantissa becomes zero. The remainders, which are noted down during the division is read in the reverse order to get the binary equivalent. This can be better illustrated using the following example.

<table>
<thead>
<tr>
<th>2</th>
<th>14</th>
<th>Remainder</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The number is written from below, that is 1110. So the binary equivalent of 14 is 1110.

If the decimal number has a fractional part, then the fractional part is converted into binary by multiplying it with 2. Only the integer of the result is noted and the fraction is repeatedly multiplied by 2 until the fractional part becomes 0. This can be explained using the following example.

0.125

\times 2

0.25 0

\times 2

0.5 0

\times 2

0 1

Here the number is written from top - 0.001. So the binary equivalent of 0.125 is 0.001.
2.3 Computer Data Code

A computer accepts data and instructions in machine language (0's and 1's form). Data must be represented internally by the bits 0 and 1. The binary coding schemes are used to represent data internally in the computer memory. In binary coding, every symbol of text data is represented by a group of bits. The group of bits used to represent a symbol is called a byte. Modern computers use 8 bits to represent a symbol. The most popular text code systems are:

(i) BCD
(ii) ASCII
(iii) EBCDIC
(iv) UNICODE

2.3.1 Binary Coded Decimal (BCD): The BCD is the simplest binary code that is used to represent a decimal number. In the BCD code, 4 bits represent a decimal number. The decimal weighing is maintained, but the digit is represented by a combination of the binary digits 0 and 1. Since ten digits 0.....9 have to be represented, a minimum of four bits must be used to encode each digit.

Hence, each digit is represented by its binary equivalent using four bits e.g., the digit 5 is equivalent to binary 0101 and the digit 9 is equivalent to binary 1001. By this method, the decimal number 59 is represented as 0101 1001. The pure binary equivalents of the decimal digits 0 through 9 are given below:

<table>
<thead>
<tr>
<th>Digit (or BCD’s)</th>
<th>Pure Binary Equivalents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
</tr>
</tbody>
</table>

The BCD equivalents of longer numbers are simply derived by appropriate juxtaposition of the equivalents in the above table. For example, the BCD equivalent of 951 is got by juxtaposing the individual equivalents of 9, 5 and 1 from this table as below:

BCD equivalent of 951: 1001 0101 0001.
This codification scheme has been extended to cover the alphabets and special symbols by adding two more bits (known as the zone bits) on the left of the 4 bit sets in the above table. By permuting the two zones bits, the representations for alphabets and special symbols can be obtained, those for the alphabets as hereunder:

<table>
<thead>
<tr>
<th>Character</th>
<th>BCD</th>
<th>Character</th>
<th>BCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00 0000</td>
<td>J</td>
<td>10 0001</td>
</tr>
<tr>
<td>1</td>
<td>00 0001</td>
<td>K</td>
<td>10 0010</td>
</tr>
<tr>
<td>2</td>
<td>00 0010</td>
<td>L</td>
<td>10 0011</td>
</tr>
<tr>
<td>3</td>
<td>00 0011</td>
<td>M</td>
<td>10 0100</td>
</tr>
<tr>
<td>4</td>
<td>00 0100</td>
<td>N</td>
<td>10 0101</td>
</tr>
<tr>
<td>5</td>
<td>00 0101</td>
<td>O</td>
<td>10 0110</td>
</tr>
<tr>
<td>6</td>
<td>00 0110</td>
<td>P</td>
<td>10 0111</td>
</tr>
<tr>
<td>7</td>
<td>00 0111</td>
<td>Q</td>
<td>10 1000</td>
</tr>
<tr>
<td>8</td>
<td>00 1000</td>
<td>R</td>
<td>10 1001</td>
</tr>
<tr>
<td>9</td>
<td>00 1001</td>
<td>S</td>
<td>01 0010</td>
</tr>
<tr>
<td>A</td>
<td>11 0001</td>
<td>T</td>
<td>01 0011</td>
</tr>
<tr>
<td>B</td>
<td>11 0010</td>
<td>U</td>
<td>01 0100</td>
</tr>
<tr>
<td>C</td>
<td>11 0011</td>
<td>V</td>
<td>01 0101</td>
</tr>
<tr>
<td>D</td>
<td>11 0100</td>
<td>W</td>
<td>01 0110</td>
</tr>
<tr>
<td>E</td>
<td>11 0101</td>
<td>X</td>
<td>01 0111</td>
</tr>
<tr>
<td>F</td>
<td>11 0110</td>
<td>Y</td>
<td>01 1000</td>
</tr>
<tr>
<td>G</td>
<td>11 0111</td>
<td>Z</td>
<td>01 1001</td>
</tr>
<tr>
<td>H</td>
<td>11 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>11 1001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

With the 6 bits, it is possible to have $2^6 = 64$ codes and this suffices to represent all the digits (10), alphabets (26, only upper case) and special symbols (around 15).

### 2.3.2. ASCII Code:
ASCII stands for American Standard Code for Information Interchange. ASCII code is used extensively in small computers, peripherals, instruments and communications devices. It has replaced many of the special codes that were previously used. It is a seven-bit code. This includes both unprintable control codes (0-31), used to control various devices in computer and printable control codes (32-127) that represents lower case and upper case letters, digits, punctuation marks, and other symbols. With 7 bits, up to 128 characters can be coded.

#### ASCII-8 Code:
A newer version of ASCII is the ASCII-8 code, which is an 8-bit code. With 8 bits, the code capacity is extended to 256 characters. This includes graphics, symbols and mathematical representations.

### 2.3.3. EBCDIC Code:
EBCDIC stands for Extended BCD Interchange Code. It is the standard character code for large computers. It is an 8-bit code without parity. With 8 bits up to 256 characters can be coded. EBCDIC has a wider range of control characters than ASCII.

In ASCII-8 and EBCDIC, the first 4 bits are known as zone bits and the remaining 4 bits represent digit values. In ASCII-7, the first 3 bits are zone bits and the remaining 4 bits
represent digit values.

Some examples of ASCII and EBCDIC values are shown in the table 2.

<table>
<thead>
<tr>
<th>Character</th>
<th>ASCII</th>
<th>EBCDIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00110000</td>
<td>11110000</td>
</tr>
<tr>
<td>1</td>
<td>00110001</td>
<td>11110001</td>
</tr>
<tr>
<td>2</td>
<td>00110010</td>
<td>11110010</td>
</tr>
<tr>
<td>3</td>
<td>00110011</td>
<td>11110011</td>
</tr>
<tr>
<td>4</td>
<td>00110100</td>
<td>11110100</td>
</tr>
<tr>
<td>5</td>
<td>00110101</td>
<td>11110101</td>
</tr>
<tr>
<td>6</td>
<td>00110110</td>
<td>11110110</td>
</tr>
<tr>
<td>7</td>
<td>00110111</td>
<td>11110111</td>
</tr>
<tr>
<td>8</td>
<td>00111000</td>
<td>11111000</td>
</tr>
<tr>
<td>9</td>
<td>00111001</td>
<td>11111001</td>
</tr>
<tr>
<td>A</td>
<td>01000001</td>
<td>11000001</td>
</tr>
<tr>
<td>B</td>
<td>01000010</td>
<td>11000010</td>
</tr>
<tr>
<td>C</td>
<td>01000011</td>
<td>11000011</td>
</tr>
<tr>
<td>D</td>
<td>01000100</td>
<td>11000100</td>
</tr>
<tr>
<td>E</td>
<td>01000101</td>
<td>11000101</td>
</tr>
<tr>
<td>F</td>
<td>01000110</td>
<td>11000110</td>
</tr>
<tr>
<td>G</td>
<td>01000111</td>
<td>11000111</td>
</tr>
<tr>
<td>H</td>
<td>01001000</td>
<td>11001000</td>
</tr>
<tr>
<td>I</td>
<td>01001001</td>
<td>11001001</td>
</tr>
<tr>
<td>J</td>
<td>01001010</td>
<td>11001010</td>
</tr>
<tr>
<td>K</td>
<td>01001011</td>
<td>11001011</td>
</tr>
<tr>
<td>L</td>
<td>01001100</td>
<td>11001100</td>
</tr>
<tr>
<td>M</td>
<td>01001101</td>
<td>11001101</td>
</tr>
<tr>
<td>N</td>
<td>01001110</td>
<td>11001110</td>
</tr>
<tr>
<td>0</td>
<td>01001111</td>
<td>11001111</td>
</tr>
<tr>
<td>P</td>
<td>01010000</td>
<td>11010000</td>
</tr>
<tr>
<td>Q</td>
<td>01010001</td>
<td>11010001</td>
</tr>
<tr>
<td>R</td>
<td>01010010</td>
<td>11010010</td>
</tr>
<tr>
<td>S</td>
<td>01010011</td>
<td>11010011</td>
</tr>
<tr>
<td>T</td>
<td>01010100</td>
<td>11010100</td>
</tr>
<tr>
<td>U</td>
<td>01010101</td>
<td>11010101</td>
</tr>
<tr>
<td>V</td>
<td>01010110</td>
<td>11010110</td>
</tr>
<tr>
<td>W</td>
<td>01010111</td>
<td>11010111</td>
</tr>
<tr>
<td>X</td>
<td>01011000</td>
<td>11011000</td>
</tr>
<tr>
<td>Y</td>
<td>01011001</td>
<td>11011001</td>
</tr>
<tr>
<td>Z</td>
<td>01011010</td>
<td>11011010</td>
</tr>
</tbody>
</table>
2.3.4 Unicode: Unicode is a worldwide character code standard. In this code system, 16-bits (2 bytes) are used to represent a single character or symbol. Using this code system, 65,536 different characters can be represented inside the computer. The first 256 codes in Unicode are identical to the 256 codes used by ASCII system.

The Unicode standard was developed in 1991 by a joint engineering team from Apple Computer Corporation and Xerox Corporation. Unicode system is supported by the popular operating systems such as Windows 2000 and OS/2 and also supported by some applications.

2.4 Bits, Bytes and Words

A bit is abbreviation of binary digit of 0 or 1. It is defined as the smallest basic unit of storage in the computer memory that has value 0 or 1 known as bit.

A byte is a basic grouping of bits (binary digits) that the computer operates on as a single unit. It consists of 8 bits and is used to represent a character by the ASCII and EBCDIC coding systems. For example, each storage location of computers using EBCDIC or ASCII-8 codes consist of electronic circuit elements or magnetic or optical media positions that can represent at least 8 bits. Thus each storage location can hold one character. The capacity of a computer’s primary storage and its secondary storage devices is usually expressed in terms of bytes.

A word is a grouping of bits (usually larger than a byte) that is transferred as a unit between primary storage and the registers of the ALU and control unit. Thus, a computer with a 32-bit word length might have registers with a capacity of 32 bits, and transfer data and instructions within the CPU in groupings of 32 bits. It should process data faster than computers with a 16-bit or 8-bit word length.

Lots of Bites: When we start talking about lots of bytes, we get into prefixes like kilo, mega and giga, as in kilobyte, megabyte and gigabyte (also shortened to K, M and G, as in Kbytes, Mbytes and Gbytes or KB, MB and GB). The following table shows the multipliers:

<table>
<thead>
<tr>
<th>Name</th>
<th>Abbr.</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kilo</td>
<td>K</td>
<td>$2^{10} = 1,024$</td>
</tr>
<tr>
<td>Mega</td>
<td>M</td>
<td>$2^{20} = 1,048,576$</td>
</tr>
<tr>
<td>Giga</td>
<td>G</td>
<td>$2^{30} = 1,073,741,824$</td>
</tr>
<tr>
<td>Tera</td>
<td>T</td>
<td>$2^{40} = 1,099,511,627,776$</td>
</tr>
<tr>
<td>Peta</td>
<td>P</td>
<td>$2^{50} = 1,125,899,906,842,624$</td>
</tr>
<tr>
<td>Exa</td>
<td>E</td>
<td>$2^{60} = 1,152,921,504,606,846,976$</td>
</tr>
<tr>
<td>Zetta</td>
<td>Z</td>
<td>$2^{70} = 1,180,591,620,717,411,303,424$</td>
</tr>
<tr>
<td>Yotta</td>
<td>Y</td>
<td>$2^{80} = 1,208,925,819,614,629,174,706,176$</td>
</tr>
</tbody>
</table>
2.8 Information Technology

We can see in this chart that kilo is about a thousand, mega is about a million, giga is about a billion, and so on. So when someone says, "This computer has a 2 giga hard drive," what he or she means is that the hard drive stores 2 gigabytes, or approximately 2 billion bytes, or exactly 2,147,483,648 bytes. Terabyte databases are fairly common these days, and there are probably a few petabyte databases floating around the world by now.

2.5 Concepts Related to Data (Data Types)

A data type is a classification of various types of data, stating the possible values for that type, the operations that can be done on that type, and the way the values of that type are stored. It is a set of data with values having predefined characteristics. Examples of data types are: integer, floating point unit number, character, string, and pointer. Usually, a limited number of such data types come built into a language.

2.5.1 Integer Number: An integer data type can hold a whole number, but no fraction. Integers may be either signed (allowing negative values) or unsigned (non-negative values only). Various sizes of integer vary from 8 bit to 64 bit word length. Byte, Word, Long and Double are the types of Integer data types.

2.5.2 Single and Double Precision: Real data values are commonly called single precision data because each real constant is stored in a single memory location. This usually gives seven significant digits for each real value. In many calculations, particularly those involving iteration or long sequences of calculations, single precision is not adequate to express the precision required. To overcome this limitation, many programming languages provide the double precision data type. Each double precision is stored in two memory locations, thus providing twice as many significant digits.

2.5.3 Logical Data Type: Use the Logical data type when we want an efficient way to store data that has only two values. Logical data is stored as true (T) or false (F)

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>Boolean value of true or false</td>
<td>1 byte</td>
<td>True (T) or False (F)</td>
</tr>
</tbody>
</table>

2.5.4 Characters: Choose the Character data type when we want to include letters, numbers, spaces, symbols, and punctuation. Character fields or variables store text information such as names, addresses, and numbers that are not used in mathematical calculations. For example, phone numbers or zip codes, though they include mostly numbers, are actually best used as Character values.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>1 byte per character</td>
<td>1 byte</td>
<td>Any characters</td>
</tr>
</tbody>
</table>

2.5.5 Strings: A data type consisting of a sequence of contiguous characters that represent the characters themselves rather than their numeric values. A String can include letters, numbers, spaces, and punctuation. The String data type can store fixed-length strings ranging in length from 0 to approximately 63K characters and dynamic strings ranging in length from 0 to approximately 2 billion characters. The codes for String characters range from 0–255. The
first 128 characters (0–127) of the character set correspond to the letters and symbols on a
standard U.S. keyboard. These first 128 characters are the same as those defined by the
ASCII character set. The second 128 characters (128–255) represent special characters, such
as letters in international alphabets, accents, currency symbols, and fractions.

2.5.6 Memo Data Type: Use the Memo data type if we need to store more than 255
characters. A Memo field can store up to 65,536 characters. If we want to store formatted text
or long documents, we can create an OLE Object field instead of a Memo field.

2.5.7 Index Fields: Index fields are used to store relevant information along with a document.
The data input to an Index Field is used to find those documents when needed. The program
provides up to twenty-five user-definable Index Fields in an Index Set.

An index field can be one of three types:
- Drop-Down Look-Up List,
- Standard,
- Auto-Complete History List.

2.5.8 Currency Fields – The currency field accepts data in dollar form by default.

2.5.9 Date Fields – The date field accepts data entered in date format.

2.5.10 Integer Fields – The integer field accepts data as a whole number.

2.5.11 Text Fields – The text field accepts data as an alpha-numeric text string.

2.6 What Is Data Processing

Data are a collection of facts - unorganized but able to be organized into useful information. A
collection of sales orders, time sheets, and class registration cards are a few examples. Data
are manipulated to produce output, such as bills and pay cheques. When this output can be
used to help people make decisions, it is called information.

Processing is a series of actions or operations that convert inputs into outputs. When we
speak of data processing, the input is data, and the output is useful information. Hence, data
processing is defined as series of actions or operations that converts data into useful
information. The data processing system is used to include the resources such as people,
procedures, and devices that are used to accomplish the processing of data for producing
desirable output.

2.6.1 Data storage hierarchy: The basic building block of data is a character, which consists
of letters (A, B, C ... Z), numeric digits (0, 1, 2 ... 9) or special characters (+, -, /, *, .1 $ ...).
These characters are put together to form a field (also called a fact, data item, or data
element). A field is a meaningful collection of related characters. It is the smallest logical data
entity that is treated as a single unit in data processing.

For example, if we are processing employees data of a company, we may have an employee
code field, an employee name field, an hours worked field, a hourly-pay-rate field, a
2.10 Information Technology

tax-rate-deduction field, etc. Fields are normally grouped together to form a record. A record, then, is a collection of related fields that are treated as a single unit. An employee record would be a collection of fields of one employee. These fields would include the employee's code, name, hours-worked, pay-rate, tax-rate-deduction, and so forth. Records are then grouped to form a file. A file is a number of related records that are treated as a unit. For example, a collection of all employee records for one company would be an employee file.

It is customary to set up a master file of permanent (and, usually, the latest) data, and to use transaction files containing data of a temporary nature. For example, the master payroll file will contain not only all the permanent details about each employee, his name and code, pay-rate, income tax rate and so forth, but it will also include the current gross-pay-to-date total and the tax paid-to-date total. The transaction payroll file will contain details of hours worked this week, normal and overtime, and, if piecework is involved, the quantity of goods made. When the payroll program is processed, both files will have to be consulted to generate this week's payslips, and the master file updated in readiness for the following week.

Figure 2.6.1: Illustrates relationship between character, field, record, and file

A data base is a collection of integrated and related master files. It is a collection of logically related data elements that may be structured in various ways to meet the multiple processing and retrieval needs of organizations and individuals. Characters, fields, records, files, and data bases form a hierarchy of data storage. In figure 2.6.1 summarizes the data storage hierarchy used by computer based processing systems. Characters are combined to make a field, fields are combined to make a record, records are combined to make a file, and files are combined to make a data base.
2.6.2 File Organizations: System designers choose to organize, access, and process records and files in different ways depending on the type of application and the needs of users. The three commonly used file organizations used in business data processing applications are sequential, direct, and indexed sequential organizations. The selection of a particular file organization depends upon the type of application. The best organization to use in a given application is the one that happens to meet the user's needs in the most effective and economical manner. In making the choice for an application, designers must evaluate the distinct strengths and weaknesses of each file organization. File organization requires the use of some key field or unique identifying value that is found in every record in the file. The key value must be unique for each record of the file because duplications would cause serious problems. In the payroll example, the employee code field may be used as the key field.

2.6.2.1 Serial File Organization: With serial file organization, records are arranged one after another, in no particular order other than, the chronological order in which records are added to the file. Serial organization is commonly found with transaction data, where records are created in a file in the order in which transactions take place. This type of file organization provides advantages like fast access to next record in sequence, cheap storage media and easy to do file back up facility. The updation can be done very slowly in this file organization. Records in a serially organized file are sometimes processed in the order in which they occur. For example, when such a file consists of daily purchase and payment transaction data, it is often used to update records in a master account file. Since transactions are in random order by key field, in order to perform this update, records must be accessed randomly from the master file.

Transaction data is not the only type of data found in serially organized files. In many businesses, customer account numbers are issued in a serial manner. In this scheme, a new customer is given the next highest account number that has not been issued and the data about the new customer (such as name, address, and phone number) are placed at the end of the existing customer account file.

<table>
<thead>
<tr>
<th>ID</th>
<th>Book Name</th>
<th>Series</th>
<th>Author</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>157</td>
<td>A Passage to India</td>
<td></td>
<td>E M Forster</td>
<td>Drama</td>
</tr>
<tr>
<td>737</td>
<td>A Private Cosmos</td>
<td></td>
<td>Philip Jose Farmer</td>
<td>Science Fiction</td>
</tr>
<tr>
<td>249</td>
<td>A Regency Scandal</td>
<td></td>
<td>Alice Chetwynd Ley</td>
<td>Historical Novel</td>
</tr>
<tr>
<td>289</td>
<td>A Sailor of Austin</td>
<td></td>
<td>John Higgins</td>
<td>Historical Novel</td>
</tr>
<tr>
<td>743</td>
<td>A Spell for Chameleon</td>
<td>1</td>
<td>Piers Anthony</td>
<td>Science Fantasy</td>
</tr>
<tr>
<td>532</td>
<td>A Stainless Steel Rat is Born</td>
<td></td>
<td>Harry Harrison</td>
<td>Science Fiction</td>
</tr>
<tr>
<td>4</td>
<td>Aboriginals of Australia</td>
<td></td>
<td></td>
<td>Anthropology</td>
</tr>
<tr>
<td>347</td>
<td>About Time</td>
<td></td>
<td>Paul Davies</td>
<td>Physics</td>
</tr>
</tbody>
</table>

Figure 2.6.2: Serial File Organization

2.6.2.2 Sequential File Organization: In a sequential file, records are stored one after another in an ascending or descending order determined by the key field of the records. In payroll example, the records of the employee file may be organized sequentially by employee code sequence. Sequentially organized files that are processed by computer systems are normally stored on storage media such as magnetic tape, punched paper tape, punched cards, or magnetic disks. To access these records, the computer must read the file in
sequence from the beginning. The first record is read and processed first, then the second record in the file sequence, and so on. To locate a particular record, the computer program must read in each record in sequence and compare its key field to the one that is needed. The retrieval search ends only when the desired key matches with the key field of the currently read record. On an average, about half the file has to be searched to retrieve the desired record from a sequential file.

<table>
<thead>
<tr>
<th>ID</th>
<th>Book Name</th>
<th>Series</th>
<th>Author</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>154</td>
<td>Complete DIY Manual</td>
<td>Readers Digest</td>
<td>DIY</td>
<td></td>
</tr>
<tr>
<td>155</td>
<td>The Prince</td>
<td>Celia Eadyfield</td>
<td>Drama</td>
<td></td>
</tr>
<tr>
<td>156</td>
<td>The Ringed Castle</td>
<td>Dorothy Duffett</td>
<td>Drama</td>
<td></td>
</tr>
<tr>
<td>157</td>
<td>A Passage to India</td>
<td>E M Forster</td>
<td>Drama</td>
<td></td>
</tr>
<tr>
<td>158</td>
<td>The Collected Short Stories of Saki</td>
<td>Hector Hugh Munro</td>
<td>Drama</td>
<td></td>
</tr>
<tr>
<td>159</td>
<td>The Pitman Dictionary</td>
<td>Isaac Pitman</td>
<td>English Language</td>
<td></td>
</tr>
</tbody>
</table>

**Advantages**

(i) Easy to organize, maintain, and understand.

(ii) There is no overhead in address generation. Locating a particular record requires only the specification of the key field.

(iii) Relatively inexpensive I/O media and devices can be used for the storage and processing of such files.

(iv) It is the most efficient and economical file organization in case of applications in which there are a large number of file records to be updated at regularly scheduled intervals. That is, when the activity ratio (the ratio of the total number of records in transaction file and the total number of records in master file) is very high. Applications such as payroll processing, billing and statement preparation, and bank cheque processing meet these conditions.

**Disadvantages**

(i) It proves to be very inefficient and uneconomical for applications in which the activity ratio is very low.

(ii) Since an entire sequential file may need to be read just to retrieve and update few records, accumulation of transactions into batches is required before processing them.

(iii) Transactions must be sorted and placed in sequence prior to processing.

(iv) Timeliness of data in the file deteriorates while batches are being accumulated.

(v) Data redundancy is typically high since the same data may be stored in several files sequenced on different keys.

**2.6.2.3 Direct Access File Organization**: Direct access file organization allows immediate direct access to individual records on the file. In direct access file organization, the records are stored and retrieved using a relative record number, which gives the position of the record.
Data Storage, Retrieval and Data Base Management Systems

in the file. This type of organization also allows the file to be accessed sequentially. The most widely used direct access techniques are depicted in the chart below:

- Direct access
  - Direct sequential Access
  - Random Access
  - Self direct addressing method (A)
  - Index sequential addressing method (B)

**Figure 2.6.4: Direct Access File Organization**

The primary storage in a CPU truly provides for direct access. There are some devices outside the CPU which can provide the direct access feature; the direct access storage devices (DASD) have the capability of directly reaching any location. Although there are several types of direct storage devices including discs and other mass storage devices, discs are by far the most widely used direct access storage devices. We will now describe the methods A and B mentioned above to show how data are stored on magnetic disks using these methods.

**Direct Sequential Access Methods**

**(A) Self (Direct) Addressing:** Under self direct addressing, a record key is used as its relative address. Therefore, we can compute the record’s address directly from the record key and the physical address of the first record in the file.

Thus, this method is suitable for determining the bucket address of fixed length records in a sequential file, and in which the keys are from a complete or almost complete range of consecutive numbers. Suppose we want to store 1,60,000 payroll records cylinder-wise in the magnetic disc pack of 6 discs. The first cylinder carries the first 800 records; the 2nd cylinder carries the next 800 records, and so on. For periodic processing of the file, the read/write heads would move cylinder by cylinder in which the records have been sequentially arranged. For example, the ten faces in the first cylinder would carry the first 800 records as below:

- \( f_1, 1 \) 1 to 80
- \( f_1, 2 \) 81 to 160
- :
- :
- :  
- \( f_1, 10 \) 721 to 800
How do we have direct access then in such a file organization? There are a total of 16,000 buckets. Let the bucket address range from 10,001 to 26,000. And the keys of the records range from 1 to 1,60,000. We wish to know where record of the key 1,49,892 is to be found i.e., in which bucket it is stored. The following arithmetic computations would have to be performed towards this purpose.

1. Divide the wanted record’s key by the number of records per bucket.
2. Add the first bucket number to the quotient to give the wanted record’s bucket:
   \[ 14989 + 10001 = 24990. \]
3. The remainder (2) is the record’s position within the bucket. The remainder 0 would indicate that it is the last record of the preceding bucket. Thus, if a manager wishes to know the qualification of a particular employee (say, no. 149892) i.e., makes a random inquiry, the above computations would be performed to derive the bucket number, command the read/write heads to move to that bucket and supply the wanted information.

But this method is highly impractical because files too have gaps in the keys and this would leave too many empty buckets i.e., storage would not be compact.

The advantage of Self-addressing is that there is no need to store an index.

The disadvantages of Self-addressing are:
(i) The records must be of fixed length.
(ii) If some records are deleted their storage space remains empty.

(B) Indexed-Sequential File Organization: The indexed sequential file organization or indexed sequential access method (ISAM), is a hybrid between sequential and direct access file organizations. The records within the file are stored sequentially but direct access to individual records is possible through an index. The type of file organization is suitable for both batch processing and online processing. Here, the records are organized in sequence for efficient processing of large batch jobs and an index is used to speed up access to the records. Indexing permit access to selected records without searching the entire file. This index is analogous to a card catalog in a library.

<table>
<thead>
<tr>
<th>Cylinder</th>
<th>Highest Record key in the Cylinder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>84</td>
</tr>
<tr>
<td>2</td>
<td>250</td>
</tr>
<tr>
<td>3</td>
<td>398</td>
</tr>
<tr>
<td>4</td>
<td>479</td>
</tr>
<tr>
<td>5</td>
<td>590</td>
</tr>
</tbody>
</table>

© The Institute of Chartered Accountants of India
To locate a record, the cylinder index is searched to find the cylinder address, and then the track index for the cylinder is searched to locate the track address of the desired record. Using Figure 2.6.5 to illustrate, we assume that the desired record has a key value of 225. The cylinder address is 2, since 225 is greater than 84 but less than 250. Then, we search the track index for cylinder 2 and find that 225 is greater than 175 and equal to 225, therefore, the track address is 4. With the cylinder address, control unit can then search through the records on track 4 within cylinder 2 to retrieve the desired records.

Advantages of indexed sequential files

(i) Permits the efficient and economical use of sequential processing techniques when the activity ratio is high.

(ii) Permits direct access processing of records in a relatively efficient way when the activity ratio is low.

Disadvantages of indexed sequential files

(i) These files must be stored on a direct-access storage device. Hence, relatively expensive hardware and software resources are required.

(ii) Access to records may be slower than direct files.

(iii) Less efficient in the use of storage space than some other alternatives.

Random Access Method: In this method, transactions can be processed in any order and written at any location through the stored file. Records are stored on disk by using a hashing algorithm. The key field is fed through hashing algorithm and a relative address is created. This address gives the position on the disk where the record is to be stored. The desired records can be directly accessed using randomizing procedure or hashing without accessing all other records in the file.

Randomizing Procedure is characterized by the fact that records are stored in such a way that there is no relationship between the keys of the adjacent records. The technique provides for
2.16 Information Technology

converting the record key number to a physical location represented by a disk address through a computational procedure.

Advantages of Direct Files

(i) The access to, and retrieval of a record is quick and direct. Any record can be located and retrieved directly in a fraction of a second without the need for a sequential search of the file.

(ii) Transactions need not be sorted and placed in sequence prior to processing.

(iii) Accumulation of transactions into batches is not required before processing them. They may be processed as and when generated.

(iv) It can also provide up-to-the-minute information in response to inquiries from simultaneously usable online stations.

(v) If required, it is also possible to process direct file records sequentially in a record key sequence.

(vi) A direct file organization is most suitable for interactive online applications such as airline or railway reservation systems, teller facility in banking applications, etc.

Disadvantages of Direct Files

(i) Address generation overhead is involved for accessing each record due to hashing function.

(ii) May be less efficient in the use of storage space than sequentially organized files.

(iii) Special security measures are necessary for online direct files that are accessible from several stations.

2.6.2.4 The Best File Organization: Several factors must be considered in determining the best file organization for a particular application. These factors are file volatility, file activity, file size, and file interrogation requirements.

(i) File volatility: It refers to the number of additions and deletions to the file in a given period of time. The payroll file for a construction company where the employee roster is constantly changing is a highly volatile file. An ISAM file would not be a good choice in this situation, since additions would have to be placed in the overflow area and constant reorganization of the file would have to occur. Other direct access methods would be better. Perhaps even sequential file organization would be appropriate if there were no interrogation requirements.

(ii) File activity: It is the proportion of master file records that are actually used or accessed in a given processing run. At one extreme is the real-time file where each transaction is processed immediately and hence at a time, only one master record is accessed. This situation obviously requires a direct access method. At the other extreme is a file, such as a payroll master file, where almost every record is accessed when the weekly payroll is processed. There, a sequentially ordered master file would be more efficient.
(iii) **File interrogation**: It refers to the retrieval of information from a file. If the retrieval of individual records must be fast to support a real-time operation such as airline reservation then some kind of direct organization is required. If, on the other hand, requirements for data can be delayed, then all the individual requests or information can be batched and run in a single processing run with a sequential file organization.

(iv) **File size**: Large files that require many individual references to records with immediate response must be organized under some type of direct access method. On the other hand, with small files, it may be more efficient to search the entire file sequentially or, with a more efficient binary search, to find an individual record than to maintain complex indexes or complex direct addressing scheme.

### 2.7 Data Base Management Systems

Traditional sequential and random files are designed to meet specific information and data processing requirements of a particular department such as accounting, sales, or purchasing etc. Different files are created to support these functions, but many of the fields on each of these files are common. For example, each of these functional areas needs to maintain customer data such as customer name, address and the person to be contracted at the customer location etc. In a traditional file environment, when information relating to any of the fields change, each relevant file must be updated separately.

Through the early 1980s, most information systems were implemented in an environment with a single functional objective (such as accounts receivable, purchase accounting, payroll etc.) in mind. The integration of information systems was not a priority. Today companies are using database management systems software (DBMS) as a tool to integrate information flow within an organization.

#### 2.7.1 What is a DBMS?

A DBMS is a set of software programs that controls the organization, storage, management, and retrieval of data in a database. A data base is a repository for related collection of data. For example, an address book can be a database where the names, address and telephone numbers of friends and business contacts are stored. A company database might contain information about customers, vendors, employees, sales and inventory. Each piece of information can be added to a data base and extracted later in a meaningful way. DBMS is the program (or collection of programs) that allows users (and other programs) to access and work with a database.

A firm might have a customer credit file containing data such as:

- Customer number
- Customer name and address
- Credit code
- Credit limit
Database programs for personal computers come in many shapes, sizes and variations. Examples of DBMS are tabulated below:

<table>
<thead>
<tr>
<th>Database</th>
<th>Manufacturer</th>
<th>Computer Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Microsoft Corporation</td>
<td>Personal computer, server, PDA</td>
</tr>
<tr>
<td>Adobe</td>
<td>Software AG</td>
<td>Midrange server, mainframe</td>
</tr>
<tr>
<td>DB2</td>
<td>IBM Corporation</td>
<td>Personal computer, midrange server</td>
</tr>
<tr>
<td>Oracle</td>
<td>Hyperion Solutions</td>
<td>Personal computer, server</td>
</tr>
<tr>
<td>FileObjects</td>
<td>FileObjects Inc</td>
<td>Personal computer, midrange server</td>
</tr>
<tr>
<td>Informix</td>
<td>IBM Corporation</td>
<td>Personal computer, mainframe</td>
</tr>
<tr>
<td>Ingres</td>
<td>Computer Associates</td>
<td>Personal computer, mainframe</td>
</tr>
<tr>
<td>InterBase</td>
<td>Sterling Software</td>
<td>Personal computer, server</td>
</tr>
<tr>
<td>Informix</td>
<td>Sterling Software</td>
<td>Personal computer, mainframe</td>
</tr>
<tr>
<td>MySQL</td>
<td>MySQL AB</td>
<td>Personal computer, midrange server</td>
</tr>
<tr>
<td>ObjectDB</td>
<td>Progress Software</td>
<td>Personal computer, midrange server</td>
</tr>
<tr>
<td>Oracle</td>
<td>Oracle Corporation</td>
<td>Personal computer, server</td>
</tr>
<tr>
<td>SQL Server</td>
<td>Microsoft Corporation</td>
<td>Server, personal computer, PDA</td>
</tr>
<tr>
<td>Sybase</td>
<td>Sybase Inc.</td>
<td>Personal computer, midrange server</td>
</tr>
<tr>
<td>Teradata</td>
<td>Teradata Corporation</td>
<td>Personal computer, midrange server</td>
</tr>
<tr>
<td>Visual FoxPro</td>
<td>Microsoft Corporation</td>
<td>Personal computer, server</td>
</tr>
</tbody>
</table>

© The Institute of Chartered Accountants of India
2.7.2 An Example of File Processing Approach: An example to illustrate why organizations started using database processing as an alternative to traditional files processing:

Another file, called a customer master file, contains:

- Customer number
- Customer name and address
- Sales region number
- Salesperson number
- Customer class
- Shipping code
- Year to date sales this year
- Year to date sales last year

A third file, for accounts receivable, contains:

- Customer number
- Customer name and address
- First invoice data
  - Invoice number
  - Invoice date
  - Invoice amount
- Second invoice data
  - Invoice number
  - Invoice date
  - Invoice amount
- nth invoice data
  - Invoice number
  - Invoice date
  - Invoice amount

Each of these files has one or more purposes. The customer credit file is used for approving customer orders, the customer master file is used for invoicing customers, and the accounts receivable file represents the money which is to be recovered from customers on account of sales by the firm. All are master files.

Some redundancies are found in the data elements contained within the files. All three files include customer number, and customer name and address. This redundancy is necessary since each file is designed to provide all of the data needed by a particular program.
Let us assume that the sales manager wants a report showing the amount of receivables by a salesperson. The firm’s customers have not been paying their bills promptly, and the sales manager wants to know which sales persons have neglected to follow up on past due receivables. He wants the report to include the data listed in Table 3. It can be seen that this special report will require data from four files. A salesperson master file is needed to provide the sales person name.

Table 3: Illustrates example of integration of report data from several files

<table>
<thead>
<tr>
<th>Report Data</th>
<th>Customer Credit File</th>
<th>Customer Master File</th>
<th>Accounts Receivable File</th>
<th>Salesperson Master File</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesperson number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales person Name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer name</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Credit code</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Year to date sales this year</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Total accounts receivable</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

The report will list each customer by salesperson, following the process illustrated in Figure 2.7.3.

In step 1, a program selects data from the three customer files that are maintained in customer number sequence. An intermediate file is created with the selected data (all the data elements listed in Table 3 except salesperson name). This intermediate file is stored into salesperson sequence in step 2. A sort is necessary since the salesperson master file is maintained in salesperson sequence. A second intermediate file is created and used with the salesperson master file to prepare the report in step 3. The programs for step 1 and step 3 would have to be specially written to satisfy this request.

Similarly a manager may require ad hoc reports for management information. For example, a manager might request a report showing sales for sales person 23. Assume that the firm assigns certain customers in a territory to a salesperson and that a customer file contains a record for each customer. The task is to select records for salesperson 23 only and print data on the report. Since the customer file is in sequence by customer, each record will have to be examined to determine if the sales person field contains a 23. This could be a time consuming process.
2.7.3 Management Problems of File Processing: For many years, information systems had a file processing orientation, as illustrated in the previous example. Data needed for each user application was stored in independent data files. Processing consisted of using separate computer programs that updated these independent data files and used them to produce the documents and reports required by each separate user application. This file processing approach is still being used, but it has several problems that limit its efficiency and effectiveness for end user applications.

(i) **Data Duplication**: Independent data files include a lot of duplicated data; the same data (such as a customer’s name and address) is recorded and stored in several files. This data redundancy causes problems when data has to be updated, since separate file maintenance programs have to be developed and coordinated to ensure that each file is properly updated. Of course, this proves difficult in practice, so a lot of inconsistency occurs among data stored in separate files.

(ii) **Lack of Data Integration**: Having data in independent files makes it difficult to provide end users with information for ad hoc requests that require accessing data stored in several different files. Special computer programs have to be written to retrieve data from each independent file. This is difficult, time consuming, and expensive for the organizations.
2.22 Information Technology

(iii) **Data Dependence**: In file processing systems, major components of a system i.e., the organization of files, their physical locations on storage, hardware and the application software used to access those files depend on one another in significant ways. For example, application programs typically contain references to the specific format of the data stored in the various files they use. Thus, if changes are made in the format and structure of data and records in a file, changes have to be made in all the programs that use this file. This program maintenance effort is a major burden of file processing systems. It is difficult to do it properly, and it results in a lot of inconsistency in the data files.

(iv) **Data Integrity and Security**: There are certain integrity constraint defined in DBMS to protect and unauthorized access to the data in the database. For example, when inserting the data for a particular field say salary for an employee database, it can not be null. Such type of constraint does not allow the user to leave the field blank thus providing integrity and security on the database. Whereas in file processing systems, such type of integrity constraint and security aspects are lacking. Also in file processing system, the integrity (i.e. the accuracy and completeness) of the data is suspected because there is no control over their use and maintenance by authorized end users.

2.7.4 The Database Management Solution: The concepts of databases and database management were, therefore, developed to solve the problems of file processing systems. A database is an integrated collection of logically related records and files. It consolidates records previously stored in independent files so that it serves as a common pool of data to be accessed by many different application programs.

Benefits of DBMS Solution are:

(i) Reduced data redundancy and inconsistency.
(ii) Enhanced data integrity and security
(iii) Provide logical and physical data independence
(iv) Provide application data independence
(v) Reduced complexity of the organization’s Information System environment.
(vi) Provide faster data accessibility and improved data sharing
(vii) Increased productivity of application development
(viii) Low cost of developing and maintaining system

2.7.5. What is a DATABASE?

Regardless of its file organization, a data base system includes several components that collectively give it certain distinct, specific characteristics. The following is a precise definition of data base as given by G.M. Scott.

*A data base is a computer file system that uses a particular file organization to facilitate rapid updating of individual records, simultaneous updating of related records, easy access to all*
records, by all applications programs, and rapid access to all stored data which must be brought together for a particular routine report or inquiry or a special purpose report or inquiry."

Each of the italicized phrases in the preceding definition has a special meaning that helps define a database. “File organization” indicates that the database has one of the three file structures (discussed in the next section) that enable programs to establish associations between the records in the database.

A database facilities “rapid updating of individual records” and “simultaneously updating of related records” that is a database permits the entry of an individual transaction to update all records affected by that transaction simultaneously. For example, consider a Rs.100,000 credit sale. In a database system, the following accounts, along with others could be updated simultaneously with the input of one transaction.

- Sales record
- Salesperson’s commissions record
- Division sales record
- Inventory item record
- Accounts receivable customer record
- Cost of sales of individual item record

If transactions are entered as they occur, records that are simultaneously updated are continuously up to date for managerial inquiry purposes. Simultaneous updating also means that the records have consistent contents. For example, the total of the sales record would be consistent with the salesperson’s commission’s record because the latter is based on the former and both are updated at the same time.

“Easy access to all records by all applications programs” means that the standard data definitions and record formats permit, for example, a payroll applications program to access the employee number, and data about them from the personnel section of the database. It also implies that work force planning programs can access pay rates from the payroll section and employees skills from the personnel section of the database. Without a database, each application program would be able to access data only from its own file.

With respect to “rapid access” to all stored data needed of a “routine report or inquiry”, routine reports can be provided quickly after the end of the accounting period and often whenever requested during the period if the processing of transactions in kept up to date. This is possible because transfer file processing is not required at the end of the period and because data summarization of reports can be fully automated within a database. In other words, little period end processing is required. Similarly, inquiries can be routinely made into the files, for example, to see whether a particular product is available for immediately shipment.
Rapid access with respect to a “special purpose report or inquiry” means that records are kept continuously up to date for unanticipated inquiries into the files by managers and that the structure of the database files facilitates the rapid development of special programs to prepare reports about unanticipated problems.

2.7.6 Architecture of a Database Management System: It follows a three level architecture—

(i) External or user view,
(ii) Conceptual or global view,
(iii) Physical or internal view.

External or user view encircles the following—

(i) It is at the highest level of the database abstraction,
(ii) It includes only those portion of database or application programs which is of concern to the users,
(iii) It is described by means of a scheme, called the external schema,
(iv) It is defined by the users or written by the programmers.

For example an external view in its Logical Record 1 may indicate employee name, employee address and in its Logical Record 2 may indicate employee name, employee address and employee code and employee salary.

Global or conceptual view, which is viewed by the Data Base Administrator, encompasses the following—

(i) All database entities and relationships among them are included,
(ii) Single view represents the entire database,
(iii) It is defined by the conceptual schema,
(iv) It describes all records, relationships and constraints or boundaries,
(v) Data description to render it independent of the physical representation.

For example a conceptual view may define employee code as a string of characters having key value, employee address also as a string, and employee salary as an integer.

The physical or internal view contains the following—

(i) It is at the lowest level of database abstraction,
(ii) It is closest to the physical storage method,
(iii) It indicates how data will be stored,
(iv) It describes data structure,
(v) It describes access methods,
(vi) It is expressed by internal schema.
The internal view instead, may define employee name is comprised of 30 characters, employee address is also comprised of 100 characters, employee code is comprised of 5 characters and employee salary is comprised of 10 numbers.

The first step in moving from ordinary file management to a database system is to separate all data definitions from the application programs and to consolidate them into a separate entity called a schema, as illustrated in the Figure 2.7.4. In addition to data definition, the schema also includes an indication of the logical relationships between various components of the database.

The schema then becomes a component of the overall database itself. From the schema, the installation can generate dictionaries containing a complete description of the database. These will, in turn, be used by systems analysts in defining new applications.

Database systems have several schemas, partitioned according to the levels of abstraction that we have discussed. At the lowest level is the physical schema; at the intermediate level is the logical schema; and at the highest level is a subschema.

2.7.7 Data Independence: It is an ability of a database to modify a schema definition at one level without affecting a schema in the next higher level. The architecture allows the features of data independence. Data independence occurs because when the schema is changed at one level, the schema at next level remains unchanged and only the mapping between the two levels is changed. Two types of Data Independence are:

- **Physical Data Independence**: To change in the internal schema, it does not need to be change in conceptual schema. Changes in internal schema may be needed because of changes in physical structure - by upgrading storage structure, to improve the performance of the system.
• **Logical Data Independence**: To change in the conceptual schema, it does not need to be change in external schema. We may change conceptual schema to expand the database by adding, deleting, updating the records in the database.

2.7.8 **Parts of DBMS**: Both the database and DBMS software is called Database System. A database system has four major parts: Data, Hardware, Software and Users, which coordinate with each other to form an effective database system.

(i) **Data**: Being an important component of the system, most organizations generate, store and process a large amount of data. The data acts as a bridge between machine parts i.e. hardware and software and the users, who access it directly or through some application programs. The data stored in the system is partitioned onto one or more databases. A database, then, is a repository for stored data. In general, it is both integrated and shared.

By integrated, it is meant that the database is a unification of several otherwise distinct data files. The individual pieces of data in the database may be shared among several different users in the sense that each of them may have access to the same piece of data. Such sharing is really a consequence of the fact that the database is integrated.

(ii) **Hardware**: The hardware consists of the secondary storage devices such as magnetic disks (hard disk, zip disk, floppy disks), optical disks (CD-ROM), magnetic tapes, etc. on which data is stored together with the I/O devices (mouse, keyboard, printers), processors, main memory, etc. for storing and retrieving the data in a fast and efficient manner. Since database can range from those of a single user with a desktop computer to those on mainframe computers with thousands of users, therefore proper care should be taken for choosing appropriate hardware devices for a required database. The hardware consists of the secondary storage volumes, disks, drums, etc. on which the database resides, together with the associated devices, control units, channels, and so forth.

![Figure 2.7.5: Simplified Picture of a Database](image)
(iii) **Software**: The software part of a DBMS acts as a bridge between user and the database. In other words, software interacts with users, application programs, and database and files system of a particular storage media (hard disk, magnetic tapes etc.) to insert, update, delete and retrieve data. For performing operations such as insertion, deletion and updation, query languages like SQL or application software like Visual Basic can be used.

(iv) **Users**: The broad classes of users are:

- **Application Programmers and System Analysts**: System analysts determine the requirements of end users; especially naive and parametric end users, and develop specifications for canned transactions that meet these requirements. Application programmers implement these specifications as programs, and than they test, debug, document, and maintain these canned transactions.

- **End Users**: These are the people who require access to the database for querying updating and generating reports. The database exists primarily for their use.

- **Database Administrator (DBA)**: DBA is responsible for authorization access to the database, for coordinating and monitoring its use, and for acquiring the needed software and hardware resources.

- **Database Designers**: These are responsible for identifying the data to be stored in the database for choosing appropriate structures to represent and store this data.

2.7.9 **Record relationship in Database**: Organizing a large database logically into records and identifying the relationships among those records are complex and time-consuming tasks. There are large number of different records that are likely to be part of a corporate database and the numerous data elements constituting those records. Further, there can be several general types of record relationships that can be represented in a database. The various types of relationships have been shown in the Figure 2.7.6.

(i) **One-to-one relationship**, as in a single parent record to a single child record or as in a husband record and wife record in a monogamous society.

(ii) **One-to-many relationships**, as in a single parent record to two or more child records – for example, a teacher who teaches three single-section courses.

(iii) **Many-to-one relationships**, as in two or more parent records to a single child record-for example, when three administrators in a small town share one minister.

(iv) **Many-to-many relationships**, as in two or more parent records to two or more child records – for example, when two or more students are enrolled in two or more courses.
2.28 Information Technology

2.8 Database Structures

A modeling language defines the schema of each database hosted in the DBMS. Database management systems (DBMS) are designed to use three database structures to provide simplistic access to information stored in databases. The optimal structure depends on the natural organization of the application's data, and on the application's requirements, which include transaction rate (speed), reliability, maintainability, scalability, and cost. There are 3
Data Storage, Retrieval and Data Base Management Systems

Database structures implemented commercially to organize records and their relationships logically. These logical organizational approaches are known as database structures. The three database structures are:

(i) Hierarchical database structure
(ii) Network database structure
(iii) Relational database structure

These models differ in the manner in which data elements (fields) can be logically related and accessed. Hierarchical models are often considered to be the most restrictive and relational models are the most flexible. Now a day object oriented model is the most popular model used in web based application development.

2.8.1 Hierarchical Database Structure: In a hierarchical database structure, records are logically organized into a hierarchy of relationships. A hierarchically structured database is arranged logically in an inverted tree pattern. For example, an equipment database, diagrammed in Figure 2.8.1 may have building records, room records, equipment records, and repair records. The database structure reflects the fact that repairs are made to equipment located in rooms that are part of buildings.

All records in hierarchy are called nodes. Each node is related to the others in a parent-child relationship. Each parent record may have one or more child records, but no child record may have more than one parent record. Thus, the hierarchical data structure implements one-to-one and one-to-many relationships.

The top parent record in the hierarchy is called the root record. In this example, building records are the root to any sequence of room, equipment, and repair records. Entrance to this hierarchy by the database management system is made through the root record i.e., building. Records that “own” other records are called parent records. For example, room records are the parents of equipment records. Room records are also children of the parent record, building. There can be many levels of node records in a database.

IBM’s IMS system and the RDM Mobile are examples of a hierarchical database system with multiple hierarchies over the same data. RDM Mobile is a newly designed embedded database...
for a mobile computer system. The hierarchical structure is used primarily today for storing geographic information and file systems.

**Features of Hierarchical Database**

(i) Hierarchically structured database are less flexible than other database structures because the hierarchy of records must be determined and implemented before a search can be conducted. In other words, the relationships between records are relatively fixed by the structure.

(ii) Ad hoc queries made by managers that require different relationships other than that are already implemented in the database may be difficult or time consuming to accomplish.

(iii) Managerial use of query language to solve the problem may require multiple searches and prove to very time consuming. Thus, analysis and planning activities, which frequently involve ad hoc management queries of the database, may not be supported as effectively by a hierarchical DBMS as they are by other database structures.

(iv) On the plus side, a hierarchical database management system usually processes structured, day-to-day operational data rapidly. In fact, the hierarchy of records is usually specifically organized to maximize the speed with which large batch operations such as payroll or sales invoices are processed.

(v) Any group of records with a natural, hierarchical relationship to one another fits nicely within the structure. However, many records have relationships that are not hierarchical.

(vi) Though a hierarchical database structure does not permit such a structure conceptually, a commercial hierarchical database management system must have ways to cope with these relationships. Unfortunately, they may not always be easy to implement.

**2.8.2 Network Database Structure:*** The network model is a variation on the hierarchical model, to the extent that it is built on the concept of multiple branches (lower-level structures) emanating from one or more nodes (higher-level structures), while the model differs from the hierarchical model in that branches can be connected to multiple nodes. The network model is able to represent redundancy in data more efficiently than in the hierarchical model.

A network database structure views all records in sets. Each set is composed of an owner record and one or more member records. This is analogous to the hierarchy's parent-children relationship. Thus, the network model implements the one-to-one and the one-to-many record structures.

However, unlike the hierarchical mode, the network model also permits a record to be a member of more than one set at one time. The network model would permit the equipment records to be the children of both the room records and the vendor records. This feature allows the network model to implement the many-to-one and the many-to-many relationship types.

Network databases generally implement the set relationships by means of pointers that directly address the location of a record on disk. This gives excellent retrieval performance, at
the expense of operations such as database loading and reorganization.

For example, suppose that in our database, it is decided to have the following records: repair vendor records for the companies that repair the equipment, equipment records for the various machines we have, and repair invoice records for the repair bills for the equipment. Suppose four repair vendors have completed repairs on equipment items 1, 2, 3, 4, 5, 7 and 8. These records might be logically organized into the sets shown in Figure 2.8.2.

![Diagram](https://via.placeholder.com/150)

**Figure 2.8.2: Example of Network Database Structure**

Notice these relationships in the above:

(i) Repair Vendor 1 record is the owner of the Repair Invoice 1 record. This is a one-to-one relationship.

(ii) Repair Vendor 2 record is the owner of the Repair Invoice 2 and 3 records. This is a one-to-many relationship.

(iii) Repair Vendor 3 record is the owner of Repair Invoice 4 and 5 records, and the Equipment 7 record owns both the Repair Invoice 5 and 6 records because it was fixed twice by different vendors. Because many equipment records can own many Repair Invoice records, these database records represent a many-to-many relationship.

(iv) Equipment 6 record does not own any records at this time because it is not required to be fixed yet.

(v) Equipment 7 and 8 own Repair Invoice 6 because the repairs to both machines were listed on the same invoice by Repair Vendor 4. This illustrates the many-to-one relationship.

Thus, all the repair records are members of more than one owner-member set: the repair vendor-repair invoice set and the equipment-repair invoice set. The network model allows us to represent one-to-one, one-to-many and many-to-many relationships. The network model also allows us to create owner records without member records. Thus, we can create and store a record about a new piece of equipment even though no repairs have been made on the equipment yet.

Unlike hierarchical data structures that require specific entrance points to find records in a hierarchy, network data structures can be entered and traversed more flexibly.
2.8.3 Relational Database Model: A third database structure is the relational database mode. Both the hierarchical and network data structures require explicit relationships, or links, between records in the database. Both structures also require that data be processed one record at a time. The relational database structure departs from both these requirements.

A relational database allows the definition of data and their structures, storage and retrieval operations and integrity constraints that can be organized in a table structure. A table is a collection of records and each record in a table contains the same fields.

Three key terms are used extensively in relational database models: relations, attributes, and domains. A relation is a table with columns and rows. The named columns of the relation are called attributes, and the domain is the set of values the attributes are allowed to take.

All relations (and, thus, tables) in a relational database have to adhere to some basic rules to qualify as relations. First, the ordering of columns is immaterial in a table. Second, there can’t be identical record in a table. And third, each record will contain a single value for each of its attributes.

A relational database contains multiple tables, with at least similar value occurring in two different records (belonging to the same table or to different tables), implies a relationship among those two records. The relationships between records in tables can also be defined explicitly, by identifying or non-identifying parent-child relationships characterized by assigning cardinality (1:1, 1:M, M:M). Tables can also have a designated single attribute or a set of attributes that can act as a “key”, which can be used to uniquely identify each record in the table.

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

2.8.3.1 Key: The word "key" is used in the context of relational database design. In pre-relational databases (hierarchical, networked) and file systems (ISAM, VSAM, etc.) "key" often referred to the specific structure and components of a linked list, chain of pointers, or other physical locator outside of the data. A key is a set of one or more columns whose combined values are unique among all occurrences in a given table. A key is the relational means of specifying uniqueness.

There are various types of relational keys.

1. Candidate Key: A Candidate key is any set of one or more columns whose combined values are unique among all occurrences (i.e., tuples or rows). Since a null value is not guaranteed to be unique, no component of a candidate key is allowed to be null.

There can be any number of candidate keys in a table. Relational pundits are not in agreement whether zero candidate keys are acceptable, since that would contradict the
(debatable) requirement that there must be a primary key.

2. **Primary Key**: The primary key of any table is any candidate key of that table which the database designer arbitrarily designates as "primary". The primary key may be selected for convenience, comprehension, performance, or any other reasons. It is entirely proper to change the selection of primary key to another candidate key.

<table>
<thead>
<tr>
<th>Enrollment</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>student</td>
<td>class</td>
</tr>
<tr>
<td>student</td>
<td>class</td>
</tr>
</tbody>
</table>

3. **Alternate Key**: The alternate keys of any table are simply those candidate keys which are not currently selected as the primary key. An alternate key is a function of all candidate keys minus the primary key.

Let's look at this simple association or join table below which holds student class enrollment:

This is the default form, and often the only form taught or recommended for a join table. The primary key of each contribution table is inherited so that this table has a compound primary key. Since the Data Definition Language (DDL) will create a unique index on the two columns, the designer knows that each student-class pair will be unique; i.e., each student may enroll in each class only once.

Several years later a new Data Base Administrator (DBA) decides that it is inefficient to use two columns for the primary key where one would do. She adds a "row id" column and makes it the primary key by loading it with a system counter. This is fine as far as an identity for each row. But now nothing prevents a student from enrolling in the same class multiple times!

<table>
<thead>
<tr>
<th>Enrollment</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>student</td>
<td>class</td>
</tr>
<tr>
<td>student</td>
<td>class</td>
</tr>
<tr>
<td>CK</td>
<td>CK</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This happened because the data model did not retain a candidate key property on the two original columns when the primary key was changed. Therefore the new DBA had no direct way of knowing (other than text notes somewhere) that these two columns must still remain unique, even though they are no longer part of the primary key.

Notice here how this could have been handled automatically by the model, if it had captured candidate keys in the first place and then generated alternate keys as a function of those candidates not in the primary key. The original two columns remain unique even after they are no longer primary.

4. **Secondary Key**: Secondary keys can be defined for each table to optimize the data access. They can refer to any column combination and they help to prevent sequential scans over the table. Like the primary key, the secondary key can consist of multiple columns.

A candidate key which is not selected as a primary key is known as secondary key.
5. Referential Integrity: A feature provided by relational database management systems (RDBMS's) that prevents users or applications from entering inconsistent data. Most RDBMS's have various referential integrity rules that we can apply when we create a relationship between two tables.

For example, suppose Table B has a foreign key that points to a field in Table A. Referential integrity would prevent from adding a record to Table B that cannot be linked to Table A. In addition, the referential integrity rules might also specify that whenever a record get deleted from Table A, any records in Table B that are linked to the deleted record will also be deleted. This is called cascading delete. Finally, the referential integrity rules could specify that whenever modification in the value of a linked field in Table A happens, all records in Table B that are linked to it will also be modified accordingly. This is called cascading update.

Figure 2.8.3: Illustrates example of Referential Integrity in the database

As discussed in previous example, consider the repair vendor relationship. The repair vendor table consists of the repair vendor master records, which contain the repair shown, in Figure 2.8.2 as network structure. This can be represented in a tabular form as shown in Figure 2.8.4. The repair vendor table consists of the repair vendor master records, which contain the repair vendor number, name and address. The table itself is really a file. Each row in the table is really a record and each column represents one type of data element.
A similar table for equipment could look like the one in Figure 2.8.5. The table contains the records for each piece of equipment in the firm. Each record also contains the number of the repair vendor who has a contract to repair that piece of equipment.

If the manager wished to create a report, showing the names of each repair vendor and the pieces of equipment that each vendor repairs, he could combine both tables into a third table. The manager might join the two tables with a query statement such as this: JOIN REPAIR VENDOR AND EQUIPMENT ON REPAIR VENDOR NUMBER.

This would create a new table with six columns: Repair Vendor Number, Repair Vendor Name, Repair Vendor Address, Equipment Number, Equipment Name and Date Purchased. Now the manager could print out only the columns for vendor name and equipment name. An example of a report is shown in figure 2.8.6.

The manager might also produce a report by selecting from both tables only the rows for specific equipment types or for equipment purchased in specific years. The important things to
notice are that the relationships, or links, do not need to be specified in advance and that whole tables or files are manipulated.

Relational databases allow the manager flexibility in conducting database queries and creating reports. Queries can be made and new tables created using all or part of the data from one or more tables. The links between data elements in a relational database do not need to be made explicit at the time the database is created since new links can be structured at any time. The relational database structure is more flexible than hierarchical or network database structures and provides the manager with a rich opportunity for ad hoc reports and queries. However, because they do not specify the relationships among data elements in advance, relational databases do not process large batch applications with the speed of hierarchical or network databases. Many relational database management system products are available. For example, Oracle and IBM offer commercial relational database management systems, Oracle and DB2 respectively.

2.9 Other Database Models

2.9.1 Distributed Database: When an organization follows a centralized system, its database is confined to a single location under the management of a single group. Sometimes an organization may require decentralizing its database by scattering it with computing resources to several locations so that running of applications programs and data processing are performed at more than one site. This is known as distributed data processing to facilitate savings in time and costs by concurrent running of application programs and data processing at various sites.

When processing is distributed since the data to be processed should be located at the processing site, the database needs to be distributed fully or partly, depending on the organizational requirements. There are two methodologies of distribution of a database.

In a replicated database duplicates of data are provided to the sites so that the sites can have frequent access to the same data concurrently. But this method of replication is costly in terms of system resources and also maintaining the consistency of the data elements.

In a partitioned database the database is divided into parts or segments that are required and appropriate for the respective sites so that only those segments are distributed without costly replication of the entire data. A database can be partitioned along functional lines or geographical lines or hierarchically.
2.9.2 Entity-Relationship Model or Database: One of the most commonly used data model is the Entity-Relationship model. An E-R model is a specialized graphic that illustrates the interrelationships between entities in a database. It is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema of a system.

The entity is defined as a distinguishable object that exists in isolation and is described by a set of attributes. An entity may be a physical object such as a house or a car, an event such as a house sale or a car service, or a concept such as a customer transaction or order. A computer, an employee, a song, a department, a city are examples of E/R Model.

A relationship is the association among several entities. For examples, a works relationship between an employee and a department, a contain relationship between an order and Item, a performs relationship between an artist and a song, and many more.

Entities and relationships can both have attributes. An attribute is a data element that describes an entity. For example: an employee entity might have a Social Security Number, Employee Name, Gross salary as an attribute of employee and department relationship.

The set of all entities or relationships of the same type is called the entity set or relationship set. The degree of relationship indicates the link between the two entities for a specified occurrence of each. The degree of relationship is also called "Cardinality". Cardinality

Figure 2.9.1: Example of distributed database

© The Institute of Chartered Accountants of India
specifies how many instances of an entity relate to one instance of another entity. Figure 2.9.2 shows the basic E-R diagram.

Figure 2.9.2: E-R Diagram

2.9.3 Object Oriented Database: It is based on the concept that the world can be modeled in terms of objects and their interactions. Objects are entities conveying some meaning for us and possess certain attributes to characterize them and interacting with each other. An Object-oriented database provides a mechanism to store complex data such as images, audio and video, etc. An object oriented database (also referred to as object-oriented database management system or OODBMS) is a set of objects. In these databases, the data is modeled and created as objects.

An object-oriented database management system (OODBMS) helps programmers make objects created in a programming language behave as a database object. Object-oriented programming is based on a series of working objects. Each object is an independently functioning application or program, assigned with a specific task or role to perform. An object-oriented database management system is a relational database designed to manage all of these independent programs, using the data produced to quickly respond to requests for information by a larger application.

In the Figure 2.9.3, the light rectangle indicates that ‘engineer’ is an object possessing attributes like ‘date of birth’, ‘address’, etc. which is interacting with another object known as ‘civil jobs’. When a civil job is commenced, it updates the ‘current job’ attribute of the object known as ‘engineer’, because ‘civil job’ sends a message to the latter object.

* www.wikipedia.org
Fig. 2.9.3: An object-oriented database design

Objects can be organized by first identifying them as a member of a class / subclass. Different objects of a particular class should possess at least one common attribute. The dark rectangles indicate ‘Engineer’ as a class and ‘Civil Engineer’ and also ‘Architect’ as both subclasses of ‘Engineer’. These subclasses possess all the attributes of ‘Engineer’ over and above each possessing at least one attribute not possessed by ‘Engineer’. The line intersecting particular object classes represents the class of structure.

Secondly, objects can be identified as a component of some other object. ‘Engineers’ are components of a ‘Civil Job Team’ which may have one to more than one number of member(s). An ‘Engineer’ may not be a member of the ‘Civil Job Team’ and may not be a member of more than one team. The dotted line intersecting particular object classes represents the part of structure. Apart from possessing attributes, objects as well as possess methods or services that are responsible for changing their states.

In the Figure 2.9.3 for example, the service ‘Experience’ as a Civil Engineer or Architect for the object ‘Engineer’ calculates how much experience the engineers of these particular two subclasses have as professionals.
The motivation for development of object-oriented analysis and design of database are encapsulation and inheritance. Encapsulation indicates that the particulars of an object are hidden in capsule to keep it apart from the other objects. In this example, only minimum details about the attributes and services of an ‘Engineer’ are exposed to other objects. But the hiding technique weakens the coupling between the objects resulting in having fewer effects when there is a change to the system. Inheritance indicates that the object in a subclass automatically acquire or inherit the attributes and services of their class. For example, the ‘Civil Engineers’ and the ‘Architects’ possess all the attributes and services of the class ‘Engineers’. In fact inheritance develops reuse of objects and higher system reliability.

The increased utilization of object-oriented programming languages like Python, Java, C#, Visual Basic, .Net, C++, Objective-C and Smalltalk have all increased the popularity of object-oriented database management system.

An object-oriented database management system is generally best used in business applications where there is a requirement for high performance processing in a complex environment. Now a day the Object oriented database system is used increasingly to store –

(i) Data about manufacturing designs in which focus is given to design objects that can be composed or decomposed into other design objects (like Telco resorts to CAD-CAM techniques),

(ii) Images, graphics, audio, video which can be used to support multimedia applications.

(iii) Data analysis reports of specialized financial services and scientific research fields.

2.9.4 Client-server Database: It is designed in a structure in which one system can connect to another system to ask it question or instruct it to perform job. The system that asks the questions and issues the instructions is the client and the system answering the queries and responding to the instructions is the server. The client machine contains the user interface logic, business logic and the database logic and the server machine contains the database. Both are coupled with a network of high bandwidth.

Figure 2.9.4: A Client-server Database Design (2-tier)
Whereas the user interface program or front end program is called the client, a back end program is called a server which interacts with share resources in an environment which can be based on heterogeneous hardware / software (Operating System) platforms of the client and the server and multi-vendor.

The above is a 2-tier model implying a complicated software distribution procedure. Since all the application logic is executed on the personal computers, all these personal computers have to be updated in case of a new software release, which is bound to be very costly, time consuming, complicated and error prone. Once it reaches one of the users, the software first has to be installed and then tested for correct execution. Due to distributed character of such a procedure, it is not assured whether all clients work on the correct copy of the program.

3-tier and n-tier client-server database designs try to solve these problems by simply transferring the application logic from the client back to the server. This is accomplished by inserting an application server tier between the data server tier and the client tier. Client tier is responsible for data presentation, receiving user events and controlling the user interface. The actual business logic is not handled by the client tier; instead it is handled by the application server tier. In the light of object-oriented analysis (OOA), business objects which implement the business rules, reside in application server tier which forms the central key to solve the 2-tier problems and protects the data from direct access by the clients. Data server tier is responsible for data storage. Besides, the relational database structure, legacy system structure is often used.

2.9.5 Knowledge Database: A database system provides functions to define, create, modify, delete and read data in a system. The type data maintained in a database system historically has been declarative data describing the static aspects of the real world objects and their associations. A pay roll file and a personnel file can share data about pay rates for each and every employee, their positions, names, etc.

A database system can also be used to maintain procedural data describing the dynamic aspects of the real world objects and their associations. The database can contain for example, several amended versions of enactments in the field of labour laws to facilitate management decisions in pay negotiations. When both the declarative and procedural data are stored in a database it constitutes a knowledge database with more powerful data maintenance.

The emergence of voluminous databases and higher use of decision support systems (DSS) and executive information systems (EIS) have led to increased interest regarding database structures which allow recognition of patterns among data and facilitate knowledge discovery by the decision makers.

A voluminous database which contains integrated data, detailed data, summarized data, historical data and metadata (data about data) is called a data warehouse. A database which contains selective data from a data warehouse meant for a specific function or department is called a data mart. The process of recognizing patterns among data contained in a data warehouse or a data mart is called a process of data mining.
2.10 Database Components

(i) **Data Definition Language (DDL)**: It defines the conceptual schema providing a link between the logical (the way the user views the data) and physical (the way in which the data is stored physically) structures of the database. As discussed earlier, the logical structure of a database is a schema. A subschema is the way a specific application views the data from the database.

Following are the functions of Data Definition Language (DDL) –

(i) They define the physical characteristics of each record, field in the record, field’s data type, field’s length, field’s logical name and also specify relationships among the records,

(ii) They describe the schema and subschema,

(iii) They indicate the keys of the record,

(iv) They provide means for associating related records or fields,

(v) They provide for data security measures,

(vi) They provide for logical and physical data independence.

(ii) **Data Manipulation Language (DML)**: –

(i) They provide the data manipulation techniques like deletion, modification, insertion, replacement, retrieval, sorting and display of data or records,

(ii) They facilitate use of relationships between the records,

(iii) They enable the user and application program to be independent of the physical data structures and database structures maintenance by allowing to process data on a logical and symbolic basis rather than on a physical location basis,

(iv) They provide for independence of programming languages by supporting several high-level procedural languages like, COBOL, PL/1 and C++.

![Figure 2.10.1: Data Base Management Systems Components](image-url)
2.11 Structure of DBMS:

(i) **DDL Compiler** –
   a. It converts data definition statements into a set of tables,
   b. Tables contain meta data (data about the data) concerning the database,
   c. It gives rise to a format that can be used by other components of database.

(ii) **Data Manager** –
    a. It is the central software component,
    b. It is referred to as the database control system,
    c. It converts operations in users’ queries to physical file system.

(iii) **File Manager** –
     a. It is responsible for file structure,
     b. It is responsible for managing the space,
     c. It is responsible for locating block containing required record,
     d. It is responsible for requesting block from disk manager,
     e. It is responsible for transmitting required record to data manager.

(iv) **Disk Manager** –
     a. It is a part of the Operating System,
     b. It carries out all physical input / output operations,
     c. It transfers block / page requested by file manager.

(v) **Query Manager** –
   a. It interprets user’s online query,
   b. It converts to an efficient series of operations,
   c. In a form it is capable of being sent to data manager,
   d. It uses data dictionary to find structure of relevant portion of database,
   e. It uses information to modify query,
   f. It prepares an optimal plan to access database for efficient data retrieval.

(vi) **Data Dictionary** –
    a. It maintains information pertaining to structure and usage of data and meta data,
    b. It is consulted by the database users to learn what each piece of data and various synonyms of data field means.
2.12 Database Administrator

As mentioned earlier database systems are typically installed and coordinated by an individual called the database administrator. A database administrator (DBA) is a person responsible for the design, implementation, maintenance and repair of an organization’s database. They are also known by the titles Database Coordinator or Database Programmer. The role includes the development and design of database strategies, monitoring and improving database performance and capacity, and planning for future expansion requirements. They may also plan, co-ordinate and implement security measures to safeguard the database. We are here to describe the roles and the responsibilities of Database Administrator which are as follows.

(i) DBA has the overall authority to establish and control data definitions and standards and also responsible for determining the relationships among data elements, and for designing the database security system to guard against unauthorised use.

(ii) DBA also trains and assists applications programmers in the use of database. A data dictionary is developed and used in a database to document and maintain the data definitions.

(iii) To design the database, the data base administrator must have a discussion with users to determine their data requirement and then decides the schedule and accuracy requirements, the way and frequency of data access, search strategies, physical storage requirements of data, level of security needed and the response time requirements.

* www.jignyasa.net
DBA may also identify the source of data and the person responsible for originating and updating of data. The database administrator then converts these requirements into a physical design that specifies hardware resources required for the purpose.

Defining the contents of the database is an important part of database creation and maintenance. The process of describing formats, relationships among various data elements and their usage is called data definition and the DBA uses data definition language (DDL) for this purpose.

Maintaining standards and controlling access to database are two other important functions that are handled by the DBA using DDL. The DBA specifies various rules which must be adhered to while describing data for a database. Data description not meeting these rules are rejected and not placed in the data dictionary. Invalid data values entered by users are also rejected. The DBA uses access controls to allow only specified users to access certain paths into the database and thus prevent unauthorized access. For example, in an airline reservation system, an airline agent should be prevented from offering an expired rate to a passenger.

The DBA also prepares documentation which includes recording the procedures, standard guidelines and data descriptions necessary for the efficient and continued use of the database environment. Documentation should be helpful to end users, application programmers, operating staff and data administration personnel. The DBA also educates these personnel about their duties.

It is also a duty of the DBA to ensure that the operating staff performs its database processing related responsibilities which include loading the database, following maintenance and security procedures, taking backups, scheduling the database for use and following restart and recovery procedures after some hardware or software failure, in a proper way.

The DBA also monitors the database environments. They ensure that the standards for database performance are being met and the accuracy, integrity and security of data is being maintained.

DBA also setup procedures for identifying and correcting violation of standards, documents and corrects errors. This is accomplished by carrying out a periodic audit of the database environment.

Finally, The DBA is responsible for incorporating any enhancements into the database environment which may include new utility programs or new systems releases, and changes into internal procedures for using data base etc.

2.13 Types of Databases

The growth of distributed processing, end user computing, decision support and executive information systems has caused the development of several types of databases. Fig 2.13.1 illustrates six of the main databases that may be found in computer using organizations.
2.46 Information Technology

Figure 2.13.1: illustrates six of the main databases

**Operational databases:** These databases store detailed data needed to support the operations of the entire organization. They are also called subject area databases (SADB), transaction databases, and production databases. Examples are a customer database, personnel database, inventory database, and other databases containing data generated by business operations.

**Management Database:** These databases store data and information extracted from selected operational and external database. They consist of summarized data and information most needed by the organization's managers and other end users. Management databases are also called information databases. These are the databases accessed by executive end-users as part of decision support systems and executive information systems to support managerial decision making.

**Information Warehouse Databases:** An information warehouse stores data from current and previous years. This is usually data that has been extracted from the various operational and management databases of an organization. It is a central source of data that has been standardized and integrated so that it can be used by managers and other end-user professionals throughout an organization. For example, an important use of information warehouse databases is pattern processing, where operational data is processed to identify key factors and trends in historical patterns of the business activity.

**End User Databases:** These databases consist of a variety of data files developed by end users at their workstations. For example, users may have their own electronic copies of documents they generated with word processing packages or received by electronic mail. Or they may have their own data files generated from spreadsheet and DBMS packages.

**External Databases:** Access to external, privately owned online databases or data banks is available, for a fee, to end users and organizations from commercial information services. Data is available in the form of statistics on economic and demographic activity from statistical
data banks. One can receive abstracts from hundreds of newspapers, magazines, and other periodicals from bibliographic data banks.

**Text Databases**: Text databases are a natural outgrowth of the use of computers to create and store documents electronically. Thus, online database services store bibliographic information such as publications in larger text databases. Text databases are also available on CD-ROM optical disks for use with microcomputer systems. Big corporations and government agencies have developed large text databases containing documents of all kinds. They use text database management systems software to help create, store, search, retrieve, modify, and assemble documents and other information stored as text data in such databases. Microcomputer versions of this software have been developed to help users manage their own text databases on CD-ROM disks.

**Image Databases**: Up to this point, we have discussed databases, which hold data in traditional alphanumeric records, and files or as documents in text databases. But a wide variety of images can also be stored electronically in image databases. For example, electronic encyclopedias are available on CD-ROM disks which store thousands of photographs and many animated video sequences as digitized images, along with thousands of pages of text. The main appeal of image databases for business users is in document image processing. Thousands of pages of business documents, such as customer correspondence, purchase orders, and invoices, as well as sales catalogues and service manuals, can be optically scanned and stored as document images on a single optical disk. Image database management software allows employees to hold millions of pages of document images. Workers can view and modify documents at their own workstations and electronically transfer them to the workstations of other end users in the organization.

**2.14 Structured Query Language and Other Query Languages**

A query language is a set of commands to create, update and access data from a database allowing users to raise adhoc queries / questions interactively without the help of programmers. Structured Query Language (SQL) is a computer programming language used to manipulate information in relational database management systems (RDBMS). SQL is both the American National Standards Institute (ANSI) and International Organization for Standardization (ISO) standard for accessing data in RDBMS. It is widely used by many database software systems, including MySQL, SQL Server™, PostgreSQL, and the Oracle® Database. While Structured Query Language is arguably easier to use than traditional computer programming languages, it is also considered to be a very powerful and often complex technology.

The SQL language is usually considered to have three parts: DML or data manipulation language, DDL or data definition language, and DCL or data control language. DML consist of SELECT, UPDATE, INSERT, and DELETE statements. DDL is made up of CREATE and ALTER statements. And DCL is comprised of GRANT and REVOKE statements. In recent years DML, has been expanded to include the MERGE statement and DDL has had the APPEND statement added.
The SQL language is sub-divided into several language elements, including:

- **Clauses**, which are in some cases optional, constituent components of statements and queries.
- **Expressions** which can produce either scalar values or tables consisting of columns and rows of data.
- **Predicates** which specify conditions that can be evaluated to Boolean (true/false/unknown) truth values and which are used to limit the effects of statements and queries, or to change program flow.
- **Queries** which retrieve data based on specific criteria.
- **Statements** which may have a persistent effect on schemas and data, or which may control transactions, program flow, connections, sessions, or diagnostics.
- SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.
- **Insignificant whitespace** is generally ignored in SQL statements and queries, making it easier to format SQL code for readability.

**Queries**: The most common operation in SQL is the query, which is performed with the declarative SELECT statement. SELECT retrieves data from one or more tables, or expressions. Standard SELECT statements have no persistent effects on the database. Some non-standard implementations of SELECT can have persistent effects, such as the SELECT INTO syntax that exists in some databases. Queries allow the user to describe desired data, leaving the database management system (DBMS) responsible for planning, optimizing, and performing the physical operations necessary to produce that result as it chooses.

A query includes a list of columns to be included in the final result immediately following the SELECT keyword. An asterisk ("*"") can also be used to specify that the query should return all columns of the queried tables. SELECT is the most complex statement in SQL, with optional keywords and clauses that include:

- The **FROM** clause which indicates the table(s) from which data is to be retrieved. The FROM clause can include optional JOIN sub clauses to specify the rules for joining tables.
- The **WHERE** clause includes a comparison predicate, which restricts the rows returned by the query. The WHERE clause eliminates all rows from the result set for which the comparison predicate does not evaluate to True.
- The **GROUP BY** clause is used to project rows having common values into a smaller set of rows. GROUP BY is often used in conjunction with SQL aggregation functions or to eliminate duplicate rows from a result set. The WHERE clause is applied before the GROUP BY clause.
• The HAVING clause includes a predicate used to filter rows resulting from the GROUP BY clause. Because it acts on the results of the GROUP BY clause, aggregation functions can be used in the HAVING clause predicate.

• The ORDER BY clause identifies which columns are used to sort the resulting data, and in which direction they should be sorted (options are ascending or descending). Without an ORDER BY clause, the order of rows returned by an SQL query is undefined.

```
SQL Statement
SELECT First Name, Last Name, MONTHLY FEE
FROM MEMBER, MEMBERSHIP PLAN
WHERE MEMBER.ID = MEMBER.MEMBERSHIP PLAN.MEMBER ID
GROUP BY FIRST NAME

SQL Statement Result
<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Monthly Fee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mario</td>
<td>Green</td>
<td>93.25</td>
</tr>
<tr>
<td>Shannon</td>
<td>Murray</td>
<td>39.50</td>
</tr>
<tr>
<td>Adrian</td>
<td>Vazquez</td>
<td>45.50</td>
</tr>
<tr>
<td>Donna</td>
<td>Vandenberg</td>
<td>55.50</td>
</tr>
<tr>
<td>Jonah</td>
<td>Weinberg</td>
<td>45.50</td>
</tr>
</tbody>
</table>
```

Figure 2.14.1: Example of SQL Statement

Some query languages have been designed in such a way that the command used is as close to Standard English text as possible. Query languages in a user-friendly way allow users to retrieve data from database without exposure to -

(i) file / record structure,

(ii) processes that the system performs,

(iii) Languages like Common Business Oriented Language (COBOL), Beginner’s All-purpose Symbolic Instruction Code (BASIC) or such other standard programming languages. Data retrieving efficiency may be improved by gaining knowledge about query shortcuts, query strategies and the types of data used in the database. A training and development program for this purpose may prove useful for the end users.

2.15 Documentation and Program Library

The documentations that are needed to support a system in an organization are–

(i) Strategic and operational plans,

(ii) Application systems and program documentation,

(iii) System software and utility program documentation,

(iv) Database documentation,

(v) Operations manuals,

(vi) User manuals,

(vii) Standard manuals.
Besides, ancillary documents like memoranda, books and journals are also the required documents to support a system. These are kept in automated form for example, computer aided systems engineering (CASE) tools are used to provide machine readable formats of dataflow diagrams or entity relationship diagrams or some software that can provide documentation on optical disks (CD-ROM) However much of the documentation is still kept in hard copy formats since it has still some advantages over online documentation.

The difficulties in the management of systems documentation are as follows –

(i) Responsibility for documentation is dispersed throughout the organization. For example, a librarian may be responsible for documentation supporting mainframe and mini computer systems, whereas documentation supporting micro computer system may be the responsibility of its users,

(ii) Documentation is maintained in multiple forms and in multiple locations. For example, some part may exist in magnetic form, some other part in hard copy form and the remaining part in the micro form,

(iii) Given the density and dispersion of documentation, proper updating, accessibility and adequate backup are not ensured.

The responsibilities of documentation librarians are ensuring that –

(i) Documentation is stored securely,

(ii) Only authorized user can have access to it,

(iii) It is updated,

(iv) Adequate backup exists for it,

Many organizations acquire a large number of software packages to support its microcomputers operations. In case the inventory of software is not managed by the documentation librarians properly, it may lead to problems like –

(i) Purchase of too many copies of the software,

(ii) Loss or theft of the software,

(iii) Loss or theft of the documentation,

(iv) Illegal copying of the software,

(v) Use of software not complying with the terms and conditions of the software license,

(vi) Absence of the software backup or improper backup.

Various types of software are available to mitigate the above hardship by taking the responsibility for maintaining the records of purchases, distributions and uses of the software and its related documentation and ensuring compliance with terms and conditions of the licensing agreements by the users. Some local area network Operation Systems for example, can provide a utility which generates a report listing all software located at workstations or file servers in the network, for review.
2.15.1 Program Library Management System Software:

(i) It provides several functional capabilities to effectively and efficiently manage data center software inventory which includes –
   a. Application Program Code,
   b. System Software Code,
   c. Job Control Statements

(ii) It possesses integrity capabilities such that –
   a. Each source program is assigned,
   b. A modification number is assigned,
   c. A version number is assigned,
   d. It is associated with a creation date

(iii) It uses –
   a. Password,
   b. Encryption,
   c. Data Compression,
   d. Automatic backup

(iv) It possesses update capabilities with the facilities of –
   a. Addition,
   b. Modification,
   c. Deletion,
   d. Re-sequencing library numbers

(v) It possesses reporting capabilities for being reviewed by the management and the end users by preparing lists of –
   a. Additions,
   b. Deletions,
   c. Modifications,
   d. Library catalogue,
   e. Library members attributes

(vi) It possesses interface capabilities with the –
   a. Operating System,
   b. Job scheduling system,
   c. Access control system,
   d. Online program management

(vii) It controls movement of program from test to production status

(viii) At last, it changes controls to application programs.
2.15.2 Design of User Interfaces: After discussing about storage media, we now turn to design of user interface. It is important since it involves the ways in which the users will interact with the system. Elements which have to considered in designing of user interface are as follows –

(i) Source documents to capture data,
(ii) Hard copy output reports,
(iii) Screen layout for source document input,
(iv) Inquiry screens for database queries,
(v) Command languages for decision support systems,
(vi) Query languages for the database,
(vii) Graphic display and color or non-monochromatic display,
(viii) Voice output to answer users or answer queries,
(ix) Screen layout manipulation by mouse or light pen,
(x) Icons for representation of the output.

The interface design is developed as follows –

(i) Identifying system users and classifying them into homogeneous groups,
(ii) The user group characteristics are understood like whether the system will be handled by novices or experts,
(iii) Eliciting the tasks which the users wish to perform using the system,
(iv) Commencing a preliminary design of the form of interaction that will support these tasks. Prototyping tools are usually employed to refine the design aspect with the users.

2.16 Backup and Recovery

Generally ‘backup and recovery’ is treated as one topic and ‘disaster recovery’ as another. ‘Backup’ is a utility program used to make a copy of the contents of database files and log files. The database files consist of a database root file, log file, mirror log file, and other database files called dbspaces.

‘Recovery’ is a sequence of tasks performed to restore a database to some point-in-time. Recovery is performed when either a hardware or media failure occurs. Hardware failure is a physical component failure in the machine, such as, a disk drive, controller card, or power supply. Media failure is the result of unexpected database error when processing data.

Before one begins recovery, it is a good practice to back up the failing database. Backing up the failing database preserves the situation, provides a safe location so files are not accidentally overridden, and if unexpected errors occur during the recovery process, database Technical Support may request these files be forwarded to them.

‘Disaster recovery' differs from a database recovery scenario because the operating system and all related software must be recovered before any database recovery can begin.
2.16.1 Database files that make up a database: Databases consist of disk files that store data. When you create a database either using any database software command-line utility, a main database file or root file is created. This main database file contains database tables, system tables, and indexes. Additional database files expand the size of the database and are called dbspaces. A dbspace contains tables and indexes, but not system tables.

A transaction log is a file that records database modifications. Database modifications consist of inserts, updates, deletes, commits, rollbacks, and database schema changes. A transaction log is not required but is recommended. The database engine uses a transaction log to apply any changes made between the most recent checkpoint and the system failure. The checkpoint ensures that all committed transactions are written to disk. During recovery the database engine must find the log file at specified location. When the transaction log file is not specifically identified then the database engine presumes that the log file is in the same directory as the database file.

A mirror log is an optional file and has a file extension of .mlg. It is a copy of a transaction log and provides additional protection against the loss of data in the event the transaction log becomes unusable.

2.16.2 Online backup, offline backup, and live backup: Database backups can be performed while the database is being actively accessed (online) or when the database is shutdown (offline). When a database goes through a normal shutdown process (the process is not being cancelled) the database engine commits the data to the database files.

An online database backup is performed by executing the command-line or from the 'Backup Database' utility. When an online backup process begins the database engine externalizes all cached data pages kept in memory to the database file(s) on disk. This process is called a checkpoint. The database engine continues recording activity in the transaction log file while the database is being backed up. The log file is backed up after the backup utility finishes backing up the database. The log file contains all of the transactions recorded since the last database backup. For this reason the log file from an online full backup must be 'applied' to the database during recovery. The log file from an offline backup does not have to participate in recovery but it may be used in recovery if a prior database backup is used.

A live backup is carried out by using the BACKUP utility with the command-line option. A live backup provides a redundant copy of the transaction log for restart of your system on a secondary machine in the event the primary database server machine becomes unusable.

Full and incremental database backup: A database backup is either a full or incremental backup. For a full backup, the database backup utility copies the database and log. An incremental backup uses the DDBACKUP utility to copy the transaction log file since the most recent full backup. When we perform an incremental backup the mirror log is not backed up. When we backup and rename the log files the transaction and mirror log file is renamed and new log files are created. We must plan to manually back up the mirror log. Be aware of this while planning out our backup and recovery strategy.
2.16.3 Developing a backup and recovery strategy: The steps suggested in the development of a backup and recovery strategy consist of the following:

- Understand what backup and recovery means to your business
- Management commits time and resources for the project
- Develop, test, time, document, health check, deploy, and monitor
- Beware of any external factors that affect recovery
- Address secondary backup issues.

(i) **Understand what backup and recovery means to your business:** How long can your business survive without access to the corporate data? Express your answer in terms of minutes, hours, or days.

If your recovery time is in minutes then database backup and recovery is critical to your business needs and it is paramount that you implement some kind of backup and recovery strategy. If recovery can take hours then you have more time to perform the tasks. If recovery can be expressed in terms of days then the urgency to recover the database still exists, but time appears to be less of a factor.

(ii) **Management commits time and resources for the project:** Management must decide to commit financial resources towards the development and implementation of a backup and recovery strategy. The strategy can be basic or quite extensive depending upon the business needs of the company. After developing a backup and recovery strategy management should be informed of the expected backup and recovery times. Anticipate management countering the timings by preparing alternative solutions. These alternative solutions could be requesting additional hardware, improved backup medium, altering backup schedule, accepting a longer recovery time versus backup time. Then it will be up to management to decide what solution fits their corporate needs.

(iii) **Develop, test, time, document, health check, deploy, and monitor:** These phases are the core in developing a backup and recovery strategy:

- Create backup and recovery commands. Verify these commands work as designed. Does your full or incremental online backup work? Verify that your commands produce the desired results.
- Time estimates from executing backup and recovery commands help to get a feel for how long will these tasks take. Use this information to identify what commands will be executed and when.
- Document the backup commands and create written procedures outlining where your backups are kept and identify the naming convention used as well as the kind of backups performed. This information can be very important when an individual must check the backups or perform a database recovery and the database administrator (DBA) is not available.
- Incorporate health checks into the backup procedures. You should check the database to ensure the database is not corrupt. You can perform a database health check prior to backing up a database or on a copy of the database from your backup.
Deployment of your backup and recovery consists of setting up your backup procedures on the production server. Verify the necessary hardware is in place and any other supporting software necessary to perform these tasks. Modify procedures to reflect the change in environment. Change user id, password, server and database name to reflect the change in environment.

- Monitor backup procedures to avoid unexpected errors. Make sure any changes in the process are reflected in the documentation.

(iv) **Beware of external factors that affect recovery**: External factors that affect database recovery are time, hardware, and software. Allow additional recovery time for entering miscellaneous tasks that must be performed. These tasks could be as simple as entering recovery commands or retrieving and loading tapes. Factors that influence time are the size of database files, recovery medium, disk space, and unexpected errors. The more files you add into the recovery scenario, it increases the places where recovery can fail. As the backup and recovery strategy develops it may be necessary to check the performance of the equipment and software ensuring it meets your expectations.

(v) **Protect database backups by performing health checks**: Database health checks are run against the database and log files to ensure they are not corrupt. The database validity utility is used to scan every record in every table and looks up each record in each index on the table. If the database file is corrupt, you need to recovery from your previous database backup. A database can be validated before being backed up or against a copy of the database from your backup.

**2.17 Data Warehouse**

A Data warehouse is a repository of an organization’s electronically stored data. Data warehouses are designed to facilitate reporting and supporting data analysis. The concept of data warehouses was introduced during the late 1980’s. The concept was introduced to meet the growing demands for management information and analysis that could not be met by operational systems. Operational systems were unable to meet this need for a range of reasons:

- The processing load of reporting reduced the response time of the operational systems,
- The database designs of operational systems were not optimized for information analysis and reporting,
- Most organizations had more than one operational system, so company-wide reporting could not be supported from a single system, and
- Development of reports in operational systems often required writing specific computer programs which was slow and expensive

As a result, separate computer databases began to be built that were specifically designed to support management information and analysis purposes. These data warehouses were able to bring in data from a range of different data sources, such as mainframe computers, minicomputers, as well as personal computers and office automation software such as spreadsheets and integrate this information in a single place. This capability, coupled with user-friendly reporting tools, and freedom from operational impacts has led to a growth of this
type of computer system. As technology improved (lower cost for more performance) and user requirements increased (faster data load cycle times and more functionality), data warehouses have evolved through several fundamental stages:

- **Offline Operational Databases** - Data warehouses in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.

- **Offline Data Warehouse** - Data warehouses in this stage of evolution are updated on a regular time cycle (usually daily, weekly or monthly) from the operational systems and the data is stored in an integrated reporting-oriented data structure.

- **Real Time Data Warehouse** - Data warehouses at this stage are updated on a transaction or event basis, every time an operational system performs a transaction (e.g. an order or a delivery or a booking etc.).

- **Integrated Data Warehouse** - Data warehouses at this stage are used to generate activity or transactions that are passed back into the operational systems for use in the daily activity of the organization.

**2.17.1 Components of a Data warehouse**: The primary components of the majority of data warehouses are shown in the Figure 2.17.1 and described in more detail below:

**Data Sources**: Data sources refer to any electronic repository of information that contains data of interest for management use or analytics. This definition covers mainframe databases (e.g. IBM DB2, ISAM, Adabas, etc.), client-server databases (e.g. Oracle database, Informix, Microsoft SQL Server etc.), PC databases (e.g. Microsoft Access), spreadsheets (e.g. Microsoft Excel) and any other electronic store of data. Data needs to be passed from these systems to the data warehouse either on a transaction-by-transaction basis for real-time data warehouses or on a regular cycle (e.g. daily or weekly) for offline data warehouses.

![Figure 2.17.1: Components of Data Warehouse](www.wikipedia.org)
Data Transformation: The Data Transformation layer receives data from the data sources, cleans and standardizes it, and loads it into the data repository. This is often called “staging” data as data often passes through a temporary database whilst it is being transformed. This activity of transforming data can be performed either by manually created code or a specific type of software could be used called an Extract, Transform and Load (ETL) tool. Regardless of the nature of the software used, the following types of activities occur during data transformation:

- Comparing data from different systems to improve data quality (e.g. Date of birth for a customer may be blank in one system but contain valid data in a second system. In this instance, the data warehouse would retain the date of birth field from the second system)
- standardizing data and codes (e.g. If one system refers to “Male” and “Female”, but a second refers to only “M” and “F”, these codes sets would need to be standardized)
- integrating data from different systems (e.g. if one system keeps orders and another stores customers, these data elements need to be linked)
- Performing other system housekeeping functions such as determining change (or “delta”) files to reduce data load times, generating or finding surrogate keys for data etc.

Data Warehouse: The data warehouse is a relational database organized to hold information in a structure that best supports reporting and analysis.

Reporting: The data in the data warehouse must be available to the organization’s staff if the data warehouse is to be useful. There are a very large number of software applications that perform this function, or reporting can be custom-developed. Examples of types of reporting tools include:

- Business intelligence tools: These are software applications that simplify the process of development and production of business reports based on data warehouse data.
- Executive information systems: These are software applications that are used to display complex business metrics and information in a graphical way to allow rapid understanding.
- Online Analytical Processing (OLAP) Tools: OLAP tools form data into logical multi-dimensional structures and allow users to select dimensions to view data.
- Data Mining: Data mining tools are software that allows users to perform detailed mathematical and statistical calculations on detailed data warehouse data to detect trends, identify patterns and analyze data.

Metadata: Metadata, or “data about data”, is used to inform operators and users of the data warehouse about its status and the information held within the data warehouse. Examples of data warehouse metadata include the most recent data load date, the business meaning of a data item and the number of users that are logged in currently.

Operations: Data warehouse operations comprises of the processes of loading, manipulating and extracting data from the data warehouse. Operations also cover user management, security, capacity management and related functions.

Optional Components: In addition, the following components also exist in some data warehouses:
1. **Dependent Data Marts**: A dependent data mart is a physical database (either on the same hardware as the data warehouse or on a separate hardware platform) that receives all its information from the data warehouse. The purpose of a Data Mart is to provide a sub-set of the data warehouse’s data for a specific purpose or to a specific sub-group of the organization.

2. **Logical Data Marts**: A logical data mart is a filtered view of the main data warehouse but does not physically exist as a separate data copy. This approach to data marts delivers the same benefits but has the additional advantages of not requiring additional (costly) disk space and it is always as current with data as the main data warehouse.

3. **Operational Data Store**: An ODS is an integrated database of operational data. Its sources include legacy systems and it contains current or near term data. An ODS may contain 30 to 60 days of information, while a data warehouse typically contains years of data. ODS’s are used in some data warehouse architectures to provide near real time reporting capability in the event that the Data Warehouse’s loading time or architecture prevents it being able to provide near real time reporting capability.

2.17.2 **Different methods of storing data in a data warehouse**: All data warehouses store their data grouped together by subject areas that reflect the general usage of the data (Customer, Product, Finance etc.) The general principle used in the majority of data warehouses is that data is stored at its most elemental level for use in reporting and information analysis. Within this generic intent, there are two primary approaches to organizing the data in a data warehouse.

The first is using a "dimensional" approach. In this style, information is stored as "facts" which are numeric or text data that capture specific data about a single transaction or event, and "dimensions" which contain reference information that allows each transaction or event to be classified in various ways. As an example, a sales transaction would be broken up into facts such as the number of products ordered, and the price paid, and dimensions such as date, customer, product, geographical location and sales person. The main advantage of a dimensional approach is it is easier to understand and use. Also, because the data is pre-processed into the dimensional form, the Data Warehouse tends to operate very quickly.

The second approach uses database normalization. In this style, the data in the data warehouse is stored in third normal form. The main advantage of this approach is that it is quite straightforward to add new information into the database, whilst the primary disadvantage of this approach is that it can be quite slow to produce information and reports.

2.17.3 **Advantages of using data warehouse**: There are many advantages to using a data warehouse, some of them are:

- Enhances end-user access to reports and analysis of information.
- Increases data consistency.
- Increases productivity and decreases computing costs.
• Is able to combine data from different sources, in one place. (Provides a common data model).
• It provides an infrastructure that could support changes to data and replication of the changed data back into the operational systems.

2.17.4 Concerns in using data warehouse
• Extracting, cleaning and loading data could be time consuming.
• Data warehouses can get outdated relatively quickly.
• Problems with compatibility with systems already in place e.g. transaction processing system.
• Providing training to end-users.
• Security could develop into a serious issue, especially if the data warehouse is web accessible.
• A data warehouse is usually not static and maintenance costs are high.

2.18 Data Mining
Data mining is concerned with the analysis of data and picking out relevant information. It is the computer, which is responsible for finding the patterns by identifying the underlying rules and features in the data.

Data mining analysis tends to work from the data up and the best techniques are those developed with an orientation towards large volumes of data, making use of as much of the collected data as possible to arrive at reliable conclusions and decisions. The analysis process starts with a set of data, uses a methodology to develop an optimal representation of the structure of the data during which time knowledge is acquired. Once knowledge has been acquired this can be extended to larger sets of data working on the assumption that the larger data set has a structure similar to the sample data. Again this is analogous to a mining operation where large amounts of low grade materials are sifted through in order to find something of value.

Examples of Data mining Software’s are SPSS, SAS, Think Analytics and G-Stat etc.

The following Figure 2.18.1 summarizes the common stages or processes identified in data mining and knowledge discovery by Usama Fayyad & Evangelos Simoudis, two of leading exponents of this area.

![Figure 2.18.1: Components of Data Mining](www.wikipedia.org)
The phases depicted start with the raw data and finish with the extracted knowledge which was acquired as a result of the following stages:

- **Selection** - selecting or segmenting the data according to some criteria e.g. all those people who own a car, in this way subsets of the data can be determined.

- **Preprocessing** - this is the data cleansing stage where certain information is removed which is deemed unnecessary and may slow down queries for e.g.: gender of the patient. The data is reconfigured to ensure a consistent format as there is a possibility of inconsistent formats because the data is drawn from several sources e.g. gender may be recorded as F or M and also as 1 or 0.

- **Transformation** - the data is not merely transferred, but transformed. E.g.: demographic overlays commonly used in market research. The data is made useable and navigable.

- **Data mining** - this stage is concerned with the extraction of patterns from the data. A pattern can be defined as given a set of facts (data) $F$, a language $L$, and some measure of certainty $C$ a pattern is a statement $S$ in $L$ that describes relationships among a subset $Fs$ of $F$ with a certainty $c$ such that $S$ is simpler in some sense than the enumeration of all the facts in $Fs$.

- **Interpretation and Evaluation** - the patterns identified by the system are interpreted into knowledge which can then be used to support human decision-making e.g. prediction and classification tasks, summarizing the contents of a database or explaining observed phenomena.

**Summary**

This chapter introduces basic concepts of database that can be used by various users to store and retrieve the data in standardized format. Various DBMS features, parts, problems and their solutions are discussed in detail. Three database structures have been defined in tune with the relationships established among various entities in the organization. Various database models which are available for designing the database for an organization are discussed in detail. The important functions of Database Administrator (DBA) and various types of databases which are useful for the organization are explained. A computer programming language such as Structured Query Language (SQL) consists of three parts namely DDL, DML & DCL and is used to create, update and access the data from a database. Data security measures have been briefly explained under Backup and recovery topic. And finally, the methods and the needs of data warehousing and data mining that are designed to facilitate reporting and supporting data analysis and extracting relevant information stored in the database have been discussed in detail.