COURSE CODE:	CSC 315
COURSE TITLE:	**File Organization & Data Processing**
NUMBER OF UNITS:	3 Units
COURSE DURATION:	Three hours per week

COURSE DETAILS:

Course Coordinator:	**Dr. O. Folorunso** B.Sc(UNAAB)., M.Sc(UNILAG)., PhD(UNAAB)
Email:	folorunsolusegun@yahoo.com, folorunsoo@unaab.edu.ng
Office Location:	Room B201, COLNAS

COURSE CONTENT:

Data processing concepts and systems, data techniques, EDP, equipment and EDP using COBOL programming output and auxiliary storage devices, types of memory access concepts of data, Physical and logical records, inter-record gaps, record structuring types and operation on files, labels, buffering blocking and deblocking, relevant i/o. Facilities for file processing of some high level programming languages such as FORTRAN, COBOL, PI/I.

COURSE REQUIREMENTS:

This is a compulsory course for all computer science students in the University. In view of this, students are expected to participate in all the course activities and have minimum of 75% attendance to be able to write the final examination.

READING LIST:

1. Linda B., Bourque, Linda B., Bourgue, Virginia A., Clark (2008). Processing Data: The Survey Example (Quantitative Applications in the Social Science), Sage Publications, ISBN 08056781901
2. Martin M. Lipschutz, Seymour Lipschutz(1989). Theory and problems of Data Processing. Mcgraw-Hill Higher Education, Singaphore.
3. French C. S (1996). Computer Science. Fifth edition, Lett Educational London.

4. Folorunso O. (2003). Essentials of Computing. Yomight Ventures, Abeokuta, Ogun State Nigeria
5. www.google.com
6. www.wikipedia.com
7. Computer Science ELBS Series by C.S. French

LECTURE NOTES

Week one

DATA PROCESSING

Electronic data processing is any process that a computer program does to enter data and summarise, analyse or otherwise convert data into usable information. The process may be automated and run on a computer. It involves recording, analysing, sorting, summarising, calculating, disseminating and storing data. Because data is most useful when well-presented and actually *informative*, data-processing systems are often referred to as information systems. Nevertheless, the terms are roughly synonymous, performing similar conversions; data-processing systems typically manipulate raw data into information, and likewise information systems typically take raw data as input to produce information as output.

Data processing may or may not be distinguished from data conversion, when the process is merely to convert data to another format, and does not involve any data manipulation.

In information processing, a **Data Processing System** is a system which processes data which has been captured and encoded in a format recognizable by the data processing system or has been created and stored by another unit of an information processing system.

DEFINITION OF DATA

Data is the basic fact about an entity. It is unprocessed information. Examples are

a. Student records which contain items like the Student Name, Number, Age, Sex etc.
b. Payroll data which contain employee number, name, department, date joined, basic salary and other allowances.
c. Driving License which contain Driver's Name, Date of Birth, Home address, Class of license and its expiry date.

Data can be regarded as the raw material from which information is produced by data processing. When data is suitably processed, results (or output data) are interpreted to derive *information* that would assist in decision- making.

DATA STORAGE UNITS ON THE COMPUTER

Data is stored in the computer in a binary form. The units used to refer to this binary data are as follows:

Term	*Definition*
Bit	The smallest unit of data storage. A bit is either a 1 or a 0.
Nibble	4 bits This term is not commonly used.
Word	This term is architecture dependent. On some systems, a word is 16 bits; on others, a word is 32 or 64 bits.
Byte (**B**)	8 bits. The most commonly used storage unit.
Kilobyte (**KB**)	Even though *kilo* usually means 1,000, a kilobyte in computer terms is actually 2^{10} or approximately 1,024 bytes (because the like of binary (2)).
Megabyte (**MB**)	The term megabyte denotes 2^{20} or 1,024KB or 1,048,576
Gigabyte (**GB**)	A gigabyte is 2^{30} or 1,024 megabytes or 1,073,741,824 bytes.
Terabyte (**TB**)	A terabyte is 2^{40} or 1,024 gigabytes or 1,099,511,627,776
Petabyte (**PB**)	A petabyte is 2^{50} or1,024 terabytes or 1,125,899,906,842,624 bytes.
Exabyte (E**B**)	A Exabyte is 2^{60} or 1,024 petabytes or 115,292,150,460,684,6976 bytes.

Fig. 1: Data storage Units used

DATA PROCESSING CYCLE

Data processing may be divided into five separate but related steps. They are:

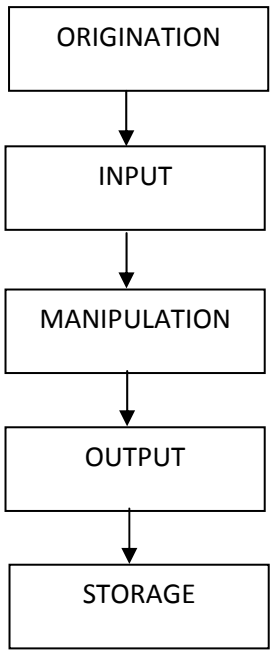


Fig. 2: Data processing Cycle

a. Origination
b. Input
c. Manipulation
d. Output
e. Storage

Origination It should be kept in mind that "to process" means to do something with or to "manipulate" existing information so that the result is meaningful and can be used by the organization to make decisions. The existing information is generally original in nature and may be handwritten or typewritten. Original documents are commonly referred to as *source documents.* Examples of source documents are cheques, attendance sheets, sales orders, invoices, receipts etc. Producing such source documents, then, is the first step in the data processing cycle.

Input After source documents are originated or made available, the next step is to introduce the information they contain into the data processing system. This system may be manual, mechanical, electromechanical or electronic. However, our focus is on electronic

data processing. This is done using any of the available input devices (keyboard, joystick etc) or data capture devices.

Processing When input data is recorded and verified, they are ready to be processed. Processing or "Manipulation" involves the actual work performed on the source data to produce meaningful results. This may require performing any or all of the following functions - *classifying, sorting, calculating, recording and summarizing.*

Output After input data has been fed into a data processing system and properly processed, the result is called output. Output can be either in summary or detail form. The type of output desired must be planned in advance so that no waste of time and resources occur. Included with output is communication. Output is of little value unless it is communicated properly and effectively. The output is the ultimate goal of data processing. The system must be capable of producing results quickly, completely and accurately. The data processing cycle is incomplete without the concept of *control.* In an organization, controls depend basically upon the comparison of attained results with predetermined goals. When the results are compared, they either agree or differ. However, if a disagreement is detected, a decision is made to make the necessary changes and the processing cycle repeated. This *feedback concept of control is* an essential part of data processing. That is, output is compared with a predetermined standard and a decision is made (if necessary) on a course of action, and is communicated to the stage where it is taken.

Storage Data related to or resulting from the previous four data processing steps can be stored, either temporarily or permanently for future reference and usage. It is necessary to store data, especially when it relates periodic reports, since they are used over and over again in other related applications. A monthly attendance report or profit and loss statements will be useful in preparing annual reports or in student result computation previous semester result will be useful in preparing present semester results in this instance requires intermittent storage. Stored information can either be raw, semi-processed or output data. Quite often, the output of one problem becomes the input to another. In the case of inventory, any unsold at the end of a year *(ending* inventory) become the *beginning* inventory for the next year. There are various ways of storing data, ranging from simple recording to storage in diskettes, hard disks, CDs etc.

ERRORS

Errors (numerical errors) An error occurs when the value used to represent some quantity is not the true value of that quantity, e.g. errors occur if we use approximate values such as 2.5 instead of 2.53627.

Note that:

a. A value may be intentionally used despite the fact that it may incur an error. The reasons for this may be those of:
   i. Simplicity
   ii. Convenience.
   iii. Cost
   iv. Necessity

b. An error may be caused by a mistake, which is said to occur if the valued use is other than the one intended, e.g. writing 5.2 when meaning to write 2.5.

OTHER ERRORS

The term "error" may also be use to describe situations that occur when a program either is not executed in the manner intended or produces results that are not correct. Causes for such errors include:

a. Faulty data
b. Faulty software.
c. Faulty hardware.

Note: The last is by far the least common

Absolute and Relative Errors

These definitions are general; computer-oriented examples will be discussed later. These are the two main types of error. They have both theoretical and practical importance.

Definitions:

a. Absolute error, This is the difference between the true value of quantity and the number used to represent it.

   I.e. absolute error = value used-True value.

   E.g. True value =2.5, value used=3

   Absolute error = 3.0 - 2.5 = 0.5

a. Relative error = $\frac{\text{Absolute error}}{\text{True value}}$

   For example, using the same figures just given

   Relative error= $\frac{0.5}{2.5} = 0.2$

   We may express the result algebraically as $E = \frac{U - T}{T}$

Where E is the relative error, U is the used value and T is the true value. The formula can be rearranged into a useful form, which is $U = T(1+E)$

Note:

i. Alternative definitions for the absolute error ignore its sign, i.e. define it in terms of numerical values or absolute values.
ii. Since the true value may not always be known, the relative error may be approximated by using:

Relative error= $\frac{\text{Absolute error estimate}}{\text{Used value}}$

For small absolute errors this gives a reasonably accurate value.

Week Two

A discussion on data and different source of error, error avoidance and reduction techniques, data processing methods. We shall also discuss different modes and processing method of data processing file accessing, organization and processing methods

Objective: The objective is for the student to understand the various validation techniques, compare and contrast and examine the advantages and disadvantages of one validation technique has over the other. Student should be able to know and implement these various validation techniques.

Description: calculation of check digit etc.

SOURCES OF ERROR

These may include followings:

a. Data errors.
b. Transcription errors.
c. Conversion errors.
d. Rounding errors.
e. Computational errors.
f. Truncation errors.
g. Algorithmic errors.

Data errors

The data input to the computer may be subject to error because of limitations on the method of data collection. These limitations may include:

i. The accuracy that it was possible to make measurements.
ii. The skill of the observer, or
iii. The resources available to obtain the data.

Transcription errors

These are errors (mistakes) in copying from one form to another.

a. Examples
   i. Transposition, eg, typing 369 for 396
   ii. "Mixed doubles", eg, typing 3226 for 3326.
b. These errors may be reduced or avoided by using
   i. Direct encoding (eg, OCR/OMR)
   ii. Validation checks.

Conversion

When converting data from its input form, BCD say, to its stored form, pure binary say, some errors may be unavoidably incurred because of practical limits to accuracy. On output similar errors may occur. Further discussion will follow later.

Rounding errors

This frequently occur when doing manual decimal arithmetic. They may occur with even greater frequency in computer arithmetic.

a. Examples.
   i. Writing 2.53627 as 2.54
   ii. Writing $^1/_3$ as 0.3333.
b. A rounding error occurs when not all the significant digits, (figures) are given e.g., when writing 2.54 we omit the less significant digits 6, 2 and 7.

Types of rounding

a. Rounding down, sometimes called truncating, involves leaving off some of the less significant digits, thus producing a lower or smaller number, e.g. writing 2.53 for 2.53627.
b. Rounding up involves leaving off some of the less significant digits, but the remaining least significant digit is increased by 1, thus making a larger number, e.g. writing 2.54 for 2.53627.
c. Rounding off involves rounding up or down according to which of these makes the least change in the stated value e.g.,2.536 would be rounded up to 2.54 but 2.533 would be rounded down to 2.53. What to do in the case of 2.535 can be decided by an arbitrary rule such as "if the next significant digit is odd round up, if even round down." So using this rule 2.535 would round to 2.54 because "3" is odd.

Significant digits (figures) and decimal places: These are the two methods of describing rounded-off results. They are defined as follows.

a. Decimal places. A number is given to n decimal places (or nD) if there are n digits to the right of the decimal point.
   Examples: 2.53627 is 2.54 to 2 decimal places i.e.
   2.53627 = 2.54 (2D)

2.53627 = 2.536 (3D)
4.203 = 4.20 (2D)
0.00351 = 0.0035 (4D)

b. Significant figures. A number is given to n significant figures (or nS) if there are n digits used to express the number but excluding

i. All leading zeros and
ii. Trailing zeros to the left of the decimal point.

Examples. 2.53627 =2-54 (3S)
57640 = 58000 (2S)
0.00351 = 0.0035 (2S)
4.203 = 4.20 (3S)

Computational errors

This occurs as a result of performing arithmetic operations and are usually caused by overflow or rounding intermediate results

Truncation errors

Firstly we need to define some terms. When numbers are placed in some specified order they form a sequence, e.g., 1, 3, 5, 7, 9….. or ½, ¼ , $^1/_8$, $^1/_{16}$, ….. When a Sequence is added it is called a **series** e.g., 1+ 3 +5 +7+9 +… or $^1/_2$ + $^1/_4$ + $^1/_8$ + $^1/_{16}$ … Some series have many practical uses. For example the quantity π, used extensively in mathematics, can be evaluated its sum, to any required accuracy be using the formula: π = 4 x (1-$^1/_3$ + $^1/_5$ - $^1/_7$ + $^1/_9$ - $^1/_{11}$ + $^1/_{13}$ ……)

The series is an **infinite** series since it goes on as far as we care to take it. In practice we might only use the first few terms to get an approximate value. We truncate a series if, when we calculate its sum, we leave off all terms past a given one, e.g.,

$$1 - {}^1/_3 + {}^1/_5 - {}^1/_7$$

is a truncated series. **A truncation** error is a resulting from the truncation of a series.

Note. *Rounding down* is also sometime called **truncation.**

Algorithmic errors

An algorithm is set of procedural steps used in the solution of a given problem and can be represented by pseudocode. Errors incurred by the execution of an algorithm are called **algorithmic errors.** A computer program is one type of algorithm. If two programs are available to perform a specified task, the one that produces the result with the greatest

accuracy will have the smaller algorithmic error. Since each step in an algorithm may involve a **computational error,** the algorithm that achieves the result in fewer steps *may* have a smaller algorithmic error.

Errors in computer arithmetic

In manual arithmetic calculations we may wish to estimate the accuracy of the data used in the calculation. The same problem exists with computer calculations, but we have to focus our attention on the errors introduced when the computer stores and handles data.

Rounding errors in stored data

Since all computers have a **finite word length** there is always a limit to the accuracy of the stores data, and many values will be rounded each time they are stored. The following factors are relevant.

a. For **fixed – point integer representation** there is good control over accuracy within the, allowed range since there is no fractional part to be rounded.
b. **In other fixed-point representations** where part or all of the number is fractional, rounding will occur often, but the precision provided may still allow reasonable control over accuracy during addition and subtraction.
c. In **floating – point representations** almost all storage and calculations can lead to rounding errors.
d. Rounding should be unbiased if possible, i.e., number should be rounded off rather than **up** or **down** when stored.

Unbiased rounding in binary

Example: Consider a very simple case where only the first two binary fraction places are available, as shown here. Consider values between $1/4$ and $1/2$.

Computer Arithmetic

Note. The number with the *third* binary place "0" are rounded down whilst those with the same place "1" are rounded up. This suggests a general rule, which is. To round off a binary fraction to n place:

If the $(n + 1)^{th}$ bit is zero, round down
If the $(n + 1)^{th}$ bit is one, round up quotation

Conversion errors
In converting fractions from decimal to binary for storage rounding errors are often introduced.

Example. $4/5$ is easily represented as the decimal fraction 0.8. However, if we convert, 0.8 to binary we discover that it can only be represented as a recurring fraction, i.e., 0.1100110011001100…. suppose we are able to store only 6 bits of this fraction, i.e., 0.110011. If we convert this store value back to decimal we will get the value 0.796875 not 0.8! Conversion errors like this are very common.

Computational errors

a. In general every arithmetic operation performed by a computer may produce a rounding error. The cause of this error will be one of:
    i. the limited number of bits available to store the result, i.e., finite word length.
    ii. Overflow or underflow (also a consequence of the first cause of this error type).
    iii. Rounding in order to normalize a result.
b. The size of the error will depend on these two main factors:
    i. The size of the word length.
    ii. The method of rounding up down or off.

Control over these errors depends on factors listed in under head of rounding errors discussed earlier.

Error avoidance and reduction

The following paragraphs outline a number of factors that either reduce errors or help in avoiding them. Detailed discussion of how these factors work is not merited but you should be able to verify the results from the examples given.

Order of operations
It is better to add "floating-point" numbers in order of magnitude if possible. For example, try calculating 0.595000 + 0.003662 + 0.000680 using only 3 digit accuracy for each intermediate result.

Algorithmic error
The errors produced when using an algorithm will frequently depend on

a. The order of the operations performed.
b. The number of operations performed.

If the error from one stage of the algorithm is carried over to successive stages then the size of the error may "grow". These **accumulated errors,** as they are called, may ultimately make the obtained very unreliable.

Nesting
This reduces the number of operations, thereby reducing error accumulation. For example to evaluate $3x^3 + 2x^2 + 5x + 1$ for a given x value use $((3x + 2) x + 5) x + 1$, starting with the innermost bracket and working outwards.

Batch adding
This is an extension of the method that will be described later. A set of numbers to be added is grouped into several batches containing numbers of similar magnitude. Each batch total is found, and then the batch totals are added.

A calculation is ill conditioned if small errors in the data used lead to large errors in the answer. **Equations** are ill conditioned if small changes in coefficients lead to large changes in the solution. Algebraic formulae as used in basic mathematics and statistics at school or college can easily become ill conditioned when certain specific values are substituted and should therefore only be used with caution. Software specially designed to solve such problems is normally based on alternative methods specially devised for the job. Such methods are relatively easy to find in suitable reference books.

DATA PROCESSING METHODS

Data originates in many different forms and they are many methods of processing: manual, mechanical and electronic. The method used, however depends on its suitability to the task at hand. There are some that are best suited for electronic processing, while others are better done by manual methods.

Manual Method

This involves preparing data by means of using such tools as pens, pencils, ledgers, files, folders etc. Improvements on these include using multi-copy forms, carbon paper etc. A good example is the daily marking of attendance register in school.

Advantages
a. They are generally cheap.

b. Simple to operate.
c. Easily adaptable to changes.
d. Easily accessible.

Disadvantages

a. May take long time to complete.
b. Cannot handle large of volume of work easily.
c. Generally prone to errors.
d. Waste a lot of manpower.

Mechanical Method

This method involves the use of a combination of manual processes and mechanical equipment to carry out the function. Examples are Typewriters, Calculators etc.

Advantages

a. Widely used in large and small organizations.
b. Can serve as input to electronic system.
c. Quality and level of output greatly improved as compared to manual method .
d. Requires less manpower than the manual method.

Disadvantages

a. Costly to purchase and maintain.
b. Possibility of equipment breakdown.
c. Produces lots of noise due to moving parts in the equipment .
d. Usually slow in operation.

Electronic Method

Here, the processing is done electronically by the system. There are two modes; batch processing and on-line processing.

Advantages

a. Faster analysis and results of processing
b. Handles complex calculations and problems
c. Can provide information in different and varied formats
d. Provides more accurate results than the other two methods
e. Work load capacity can be increased easily without hitches
f. Provides for standardization of method
g. Frees staff from clerical tasks for other tasks e.g. planning

Disadvantages

a. Initial acquisition cost may be high as well as maintenance costs
b. Specialist personnel may be required
c. Decreased flexibility as tasks become standards

PROCESSING MODES OF DATA PROCESSING

There are two modes of computer data processing; **Batch Processing and On-line Processing.**

Batch Processing

A method of processing information in which transactions are accumulated and stored until a specified time when it is necessary or convenient to process them as a group is called Batch Processing. This method is usually adopted in payroll processing and sales ledger updates.

On-line Processing

A method of processing information in which, transactions are entered directly into the computer and processed immediately. The on-line method can take different forms. These forms are examined below.

Real Time Processing This is an on-line processing technique in which a transaction undergoes all the data processing stages immediately on data capture. This method is used in Airline ticket reservation and modern retail banking software.

Multiprogramming - This method permits multiple programs to share a computer system's resources at any one time through the concurrent use of the CPU. By concurrent use, we mean that only one program is actually using the CPU at any given moment, but that the input/output needs of other programs can be serviced at the same time. Two or more programs are active at the same time, but they do not use the same computer resources simultaneously. With multiprogramming, a set of programs takes turns using the processor.

Multitasking - This refers to multiprogramming on single-user operating system such as those in microcomputers. One person can run two or more programs concurrently on a single computer. For example, the user can be working on a word-processing program and at the same time be doing a search on a database of clients. Instead of terminating the session with the word processing program, returning to the operating system, and then initiating a session with the database program, multitasking allows the display of both programs on the computer screen and allows the user to work with them at the same time.

Time Sharing - This capability allows many users to share computer-processing resources simultaneously. It differs from multiprogramming in that the CPU spends a fixed amount of time on one program before moving on to another. In a time-sharing environment, the different users are each allocated a time slice of computer time. In this time slot, each user is free to perform any required operations; at the end of the period, another user is given a time slice of the CPU. This arrangement permits many users to be connected to a CPU simultaneously, with each receiving only a tiny amount of CPU time. Time-sharing is also known as interactive processing. This enables many users to gain an on-line access to the CPU at the same time, while the CPU allocates time to each user, as if he is the only one using the computer.

Virtual Storage - Virtual storage was developed after some problems of multiprogramming became apparent. It handles programs more efficiently because the computer divides the programs into small fixed or variable length portions, storing only a small portion of the program in primary memory at one time, due to memory size constraints as compared program needs. Virtual storage breaks a program into a number of fixed-length portions called **pages** or variable length portions called **segments.** The programmer or the operating system determines the actual breakpoint between pages and segments. All other program pages are stored on a disk unit until they are ready for execution and then loaded into primary memory. Virtual storage has a number of advantages. First, primary storage is utilized more fully. Many more programs can be in primary storage because only one page of each program actually resides there. Secondly, programmers need not worry about the size of the primary storage area. With virtual storage, there is no limit to a program's storage requirements

Week Three

A discussion on data and different validation techniques, for both on-line and batch systems of processing data. We shall also discuss data hierarchy, different file accessing, organization and processing methods

Objective: The objective is for the student to understand the various validation techniques, compare and contrast and examine the advantages and disadvantages of one validation technique has over the other. Student should be able to know and implement these various validation techniques.

Description: calculation of check digit etc.

DATA VALIDATION TECHNIQUES

GIGO stands for Garbage-In, Garbage-Out. This means that whatever data you pass or enter into the computer system is what would be processed. The computer is a machine and therefore has no means of knowing whether the data supplied is the right one or not. To minimize such situations that may lead to the computer processing wrong data and producing erroneous output, data entered into a computer is validated within specific criteria to check for correctness before being processed by the system. This process is called **DATA VALIDATION.** We stated above that computer data processing is done in batch and **on-line processing modes** and we shall therefore discuss data validation techniques under each of these two modes.

Batch Control

This type of input control requires the counting of transactions or any selected quantity field in a batch of transactions prior to processing for comparison and reconciliation after processing. Also, all input forms should be clearly identified with the appropriate application name and transaction type (e.g. Deposits, Withdrawals etc). In addition, pre-numbered and pre-printed forms can be used where constant data are already printed and used to reduce data entry or recording errors.

Types of Batch Controls include:

- **Total Monetary Amount -** This is used to verify that the total monetary value of items processed equals the total monetary value of the batch documents.
- **Total Items -** This verifies that the total number of items included on each document in the batch agrees to the total number of items processed. For example, the total number of items in the batch must equal the total number of items processed.
- **Total Documents -** This verifies that the total number of documents in the batch equals the total number of documents processed. For example, the total number of invoices agrees with the number of invoices processed.
- **Hash Total -** Hashing is the process of assigning a value to represent some original data string. The value is known as hash total. Hashing provides an efficient method of checking the validity of data by removing the need for the system to compare the actual data, but instead allowing them to compare the value of the hash, known as the hash total, to determine if the data is same or different. For example, totals are obtained on an identifier (meaningless) data fields such as account number, part number or employee number. These totals have no significance other than for internal system control purposes. The hash total is entered at the start of the input

process; after completion, the system re-calculates this hash total using the selected fields (e.g. account number) and compares the entered and calculated hash total. If the same, the batch is accepted or otherwise rejected.

On-Line Transactions Control

An advantage of on-line real time systems is that data editing and validation can be done up front, before any processing occurs. As each transaction is input and entered it can be operator prompted immediately an error is found and the system can be designed to reject additional input until the error is corrected. The most important data edit and validation techniques are discussed below, but the list is by no means exhaustive.

- **Reasonableness Check -** Data must fall within certain limits set in advance or they will be rejected. For example, If an order transaction is for 20,000 units and normally not more than 100 units, then the transaction will be rejected.
- **Range Check -** Data must fall within a predetermined range of values. For example, if a human weighs more than 150kg, the data would be rejected for further verification and authorization.
- **Existence Check -** Data are entered correctly and agree with valid predetermined criteria. For example, the computer compares input reference data like Product type to tables or master files to make sure the codes are valid.
- **Check Digit -** An extra reference number called *a check digit follows an* identification code and bears a mathematical relationship to the other digits. This extra digit is input with the data, recomputed by the computer and the result compared with the one entered.

Modulo Numbers and check Digits Calculation

1. i. A check digit is a means of ensuring that a number (e.g. a customer account number) maintains it validity.

   ii. It is calculated using a modulus are used in practice and each had varying degrees of success at preventing certain types of errors MODULUS 11 (eleven) is used here.

2. Modulus notation. Two numbers are congruent in a modulus if both yield the same reminder
   When divided by the modulus
   means congruent to
   ↓
   E.g. 8 ≡3 (mod 5)

i.e., has remainder 3 if division by 5, and so does 3

3 Finding modulus 11 check digits requires the division by 11 to find a reminder, e.g., 15≡4 (mod 11).

CALCULATIONS

4. Check digits are calculated by a computer in the first place and are generally used in conjunction with fixed data (i.e., customers' number, etc). As a result of a test done on modulus 11 it was discovered that is detected all transcription and transposition errors and 91% of random errors.

5. Calculating the check digit
   i. Original code number **6349**
   ii. Multiply each digit by the yield **5432**
   iii. Product = (6x5) = 30
   (3x4) = 12
   (4x3) = 12
   (9x2) = 18

The yield gives a weighting factor for each digit in the original number.

iv. Sum of products 30+12+12+18 =**72**
v. division by 11 = 6, remainder 6, i.e. **72≡ 6 (mod 11)**
vi. Subtract remainder from modulus **(i.e. 11-6)**=5
vii. 5 is the check digit.
viii. Code number now becomes **63495.**
ix. If the number yields a check digit greater than 9 it may be discarded or the check digit replaced by some other value.

6. Checking numbers. When the code number is input to the computer precisely the same calculation can be carried out (using weight of 1 for the rightmost digit) and the resultant remainder should be 0. If not, then the number is incorrect.
   63495 = (6x5) + (3x4) + (4x3) + (9x2) + (5x1) = 77 Divide by 11, remainder = 0
7. This check can be carried out off-line by a machine called CHECK DIGIT VERIFIER
8. This is a useful programming exercise and it may also be worth including an examination project in which account numbers or similar keys were used.
9. Check digits are used in many situations. The ISBN number in any book (see the back cover page of a textbook) is just one example.

- **Completeness Check -** A field should always contain data and riot zeros or blanks. A check of the field is performed to ensure that some form of data, not blanks or zeros is present. For example, employee number should not be left blank as it identifies that employee in the employee record.

- **Validity Check -** This is the programmed checking of data validity in accordance with predetermined criteria. For example, a gender field should contain only M(ale) or F(emale). Any other entry should be rejected.

- **Table Lookups -** Input data complies with predetermined criteria maintained in a computerized table of values. For example, a table maintains the code for each local government in the country and any number entered must correspond to codes found in the table.

- **Key Verification -** another individual using a program that compares the original entry to the repeated keyed input repeats the key-in process. For example, the account number, date and amount on a cheque is keyed in twice and compared to verify the keying process.

- **Duplicate Check -** New transactions are matched to those previously entered. For example, an invoice number is checked to ensure that it is not the same as those previously entered, so that payment is made twice.
- **Logical Relationship Check -** If a particular condition is true, then one or more additional conditions or data input relationship might be required to be true before the input can be considered valid. For example, an employee applying to be paid maternity leave allowance may be required to be at least eighteen years from date of birth or employment date and be a female employee.

Week Four

Data hierarchy , file accessing (sequential, direct, index sequential and object oriented file access),flat file database file, file processing (updating, sorting, merging, blocking, searching and matching), physical storage consideration, initialization, formatting, defragmentation method.

Objective:

- To explore programming language constructs that support data abstraction and
- To discuss the general concept of file access and processing in data processing.

- Impact to student the knowledge required in choosing appropriate file access and processing technique when developing data processing application software.

Description: Data constituent in their hierarchy are discussed in detail and storage preparatory requirement of storage devices need were emphasized.

The Data Hierarchy

A computer system organizes data in a hierarchy that starts with bits and bytes and progresses to fields, records, files, and databases.

Bit: This represents the smallest unit of data a computer can handle. A group of bits, called a byte, represents a single character, which can be a letter, number or other symbol.

Field: A field is a particular place in a record where an item of information can be held or a grouping of characters into a **word,** group of words or a complete number (e.g. a person's first name or age), is called a **field**.

Record: A group of related fields, such as a student's name, class, date admitted, age or record is a collection of related items of data treated as a unit.

File: A file is organized collection of related records which are processed together. It is also referred to as a data set. The files is a collection of records relating to some class of object e.g. records of all insurance policies issue by an insurance company, records of all employees of firm, student records etc. A group of records of the same type (e.g. the records of all students in the class) is called a ***file*.**

Database: A group of related files (e.g. the personal history, examinations records and payments history files) make up a database. A record describes an entity. An entity is a person, place, thing, or event on which we maintain information. An employee record is an entity in a personnel records file and maintains information on the employees in that organization. Each characteristic or quality describing a particular entity is called an **attribute.** For example, employee name, address, age, gender, date employed is an attribute each of the entity personnel. The specific values that these attributes can have can be found in the field of the record describing the entity. Every record in the file contains at least one field that uniquely identifies that record so that the record can be retrieved, changed, modified or sorted. This identifier is called the key field. An example of a key field is the employee number for a personnel record containing

employee data such as name, address, age, job title etc.

File Accessing Methods

Computer systems store files in secondary storage (e.g. hard disks) devices. The records can be arranged in several ways on the storage media, and the arrangement determines the manner in which the individual records can be accessed or retrieved.

Sequential Access File Organization - In sequential file organization, data records must be retrieved in the same physical sequence in which they are stored. Sequential file organization is the only method that can be used on magnetic tape.(e.g. data or audio tape). This method is used when large volumes of records are involved and it is suitable for batch -processing as it is slow.

Direct/Random Access File Organization - This is a method of storing records so that they accessed in any sequence without regard to their actual physical order on the storage media. This method permits data to be read from and written back to, the same location. The physical location of the record in the file can be computed from the record key and the physical address of the first record in the file, using a *transform algorithm,* not an index. (The transform algorithm is a mathematical formula used to translate the key field directly into the record's physical location on disk.) Random access file organization is good for large files when the volume of transactions to be processed against the file is low. It is used to identify and update an individual's record on a real-time basis. It is fast and suitable for on-line processing where many searches for data are required. It is faster than sequential file access method. An example is an on-line hotel reservation system.

Index Sequential Access Method (ISAM) - This file access method directly accesses records organized sequentially using an index of key fields. An index to a file is similar to the index of a book, as it lists the key fields of each record and where that record is physically located in storage to ensure speedy location of that record. ISAM is employed in applications that require sequential processing of large numbers of records but occasionally require direct access of individual records. An example is in airline reservation systems where booking can be taking place in different parts of the world at the same time accessing information from one file. ISAM allows access to record in the most efficient manner.

Flat File - Supports a batch-processed file where each record contains the same type of data elements in the same order, with each data element needing the same number of storage spaces. Supports a few users' needs. It is inflexible to changes. It is used to enter data into an

application automatically in a batch mode, instead of record by record. This process of automatic batch data entry is also referred to as a File Upload process.

Database File - A database supports multiple- users needs. The records are related to each other differently for each file structure. Removes the disadvantages of flat files.

Object Oriented File Access - Here, the application program accesses data objects and uses a separate method to translate to and from the physical format of the object.

File Processing

Different processes can be performed on files stored in the computer system. These processes include:

- **Updating -** The process of bringing information contained in the file up to date by feeding in current information.
- **Sorting -** Arranging the records in a file in a particular order (e.g. in alphabetical or numerical order within a specified field).
- **Merging -** Appending or integrating two or more files into a bigger file.
- **Blocking -** This is to logically arrange the records in a file into fixed or variable. blocks or sets that can be treated as a single record at a time during processing. The gap between each block is known as the inter-block gap..
- **Searching -** This involves going through a whole file to locate a particular record or a set of records, using the key field.
- **Matching -** This involves going through a whole file to locate a particular record or a set of records, using one or a combination of the file attributes or fields.

Physical storage consideration

"Volume" is a general term for any individual physical storage medium that can be written to or read from. Examples include: a fixed hard disk, a disk pack, a floppy disk, a CD-ROM, a disk cartridge or a tape cartridge.

Initialization: Before a disk may be recorded upon it normally has to be initialized which involves writing zeroes to every data byte on every track. A special program is normally supplied for this purpose. Clearly, the re-initialization of a disk effectively eliminates all trace of any existing data.

Formatting: In addition to Initialization the disk has to be formatted which means that a regular pattern of blank sectors is written onto the tracks. In the case of floppy disks the "formatting" program normally combines formatting with Initialization. On magnetic tapes the format is defined when the tape is mounted on the drive. Blocks of data are then formatted as they are written to the tape. The format determines the effective storage capacity of the volume. For example, a double sided floppy disk with 80 tracks per side and 9 sectors per track with each sector containing 512 data bytes will have a storage capacity of 720 Kbytes (i.e., 9 x 40 x 2 x 512 bytes). Formats depend upon the manufacturer and operating system used. If data is to be transferred from one computer to another not only must be the volume physically interchangeable between drives the volume format must be compatible too. In present day floppy disks is out of usage but Compact Disk (CD) and Digital Video Disk are the present time usage these also requires the initialization and formatting as well though during its writing process under the control of the Operating System or specialize disk writer software installed.

Disk defragmentation.

Fragmentation: As data is stored on a newly formatted disk the data is written to unused contiguous sectors (i.e., those sectors which follow one another). If data is erased then the deleted sectors may leave "holes" of free space among used sectors. Over time, after many inserts and deletes, these free sectors may be scattered across the disk so that there may be very little contiguous free space. This phenomenon is called "disk fragmentation". If a file, such as a document file say, is written to the disk the read-write heads will have to move about as they access the fragmented free space. This slows down the writing process and will also slow down any subsequent reads. Therefore, performance suffers. When this happens it may be possible to use a special disk defragmenter program to re-organise the data on the disk so as to eliminate the fragmentation.

a. Auxiliary storage is used, as its name suggests, to supplement main storage.

b. The descriptions have concentrated on the physical features rather than uses.
c. The main auxiliary storage media are in *magnetic* form.
d. The main hardware devices and media for auxiliary storage are
   i. Magnetic disk unit - magnetic disk.
   ii. Magnetic diskette unit - magnetic diskette (floppy disk).
   iii. Optical disk unit - optical disk.
   iv. Magnetic tape unit - magnetic tape.

The comparative performance of backing storage media and devices is shown

Devices And Media.	Typical Access Time.	Typical Storage Capacities.	Typical i transfer Rates.	Types of Storage SAS or DAS	Where used as primary medium.
1. Floppy (diskette)	260ms	180 K bytes to 1.25 M bytes	24,000 bps - 50,000 bps (bytes per second)	DAS	Small micro computer systems - otherwise as a back
2. Magnetic Disk	20 - 60 ms	60 Mbytes - 5 Gbytes	312,000 bps - 2,000,000 bps	DAS	Minicompute and
3. Optical Disk	Looms	55 Mbytes - 10 Gbytes	200,000 bps	DAS	Minicompute and mainframes - for
4. Magnetic Tape (reel-to-reel)	A search is required.	40 Mbytes - 160 Mbytes	160,000 bps - 1.250.000 bps	SAS	Minicompute and - mostly as a back-up
5. Magnetic tape cartridge	A search is required.	50 Mbytes - 10 Gbytes	160,000 bps - 2.6 Mbps		Microcomput And minicompute
6. Magnetic tape cassette	A search is required.	Up to 145,000 bytes.	10 bps - 33,000 bps	SAS	Small micro Computer systems.

Fig. 4: Comparative performance of backing storage media and devices.

Points to note

a. Note the terms "on-line" and "off-line". "On-line" means being accessible to and under the control of the processor. Conversely, "off-line" means *not* accessible to or under the control of the processor. Thus, *fixed* magnetic disks *are permanently* "on-line"; a magnetic tape reel or an exchangeable magnetic disk pack is "on-line" when placed in its respective units, but "off-line" when stored away from the computer, terminals, *wherever* their physical location, are said to be "on-line" when directly linked to the processor.

b. On exchangeable disks, the read-write heads serving each *surface will* be positioned over the *same relative track* on *each surface* because the arms on which they are fixed move simultaneously.

c. The "jukebox" was introduced in this segment as an option used with CDs but jukeboxes are available for a variety of disk devices.

d. The term "*cartridge*" *is* ambiguous unless prefixed by "*tape*" or "*disk*".

e. The devices and media have been separated for ease of explanation. It should be noted, however, that in the case of the *fixed* disk the media are permanently fixed to the device, and therefore they cannot be separated.

f. Backing storage is also called **auxiliary storage.**

g. Input, output and storage devices are referred to collectively as peripheral devices.

Week Five Lecture Note

Terminals and workstations, printers and there features, Actuators

Objective:

- To discuss the general concept of distributed systems approach to data processing.
- Impact to student the knowledge of processing in distributed system environment.
- To discuss the techniques use by various printer types in printing hardcopy output
- To have know how required in choosing appropriate printer for output design

Description: Distributed processing and printers with the kind of hardcopy output quality generated are discussed in detail and features of the printers were emphasized.

Terminals and Workstations

All these devices are "keyboard devices", which merely means that their primary means of entering data to the computer is via a keyboard. The keyboards resemble the QWERTY typewriter keyboard, but usually have several additional keys, which are used for other purposes depending upon the type of device.

Terminals

Ever since inception of computer the most common form of **terminal** is the VDU. Terminals have a keyboard for input and a display screen or printer to show both what is typed in and what is output by the computer. Terminals which have a printer instead of a screen are called "terminal typewriters". They are now rather outdated and rare, so the name "terminal" is now normally synonymous with VDU and is often used instead. There are many different types of VDU terminals in use today. Only the more common features and variants will be described.

Features of the **VDU Terminal**

a. It is a dual-purpose device with a keyboard for data input and a cathode ray tube screen for output. The latter is similar to a TV screen. The screen is normally housed along with the device's electronic components in a cabinet similar to that used for a PC's monitor.
b. The keyboard resembles the QWERTY typewriter keyboard, but usually has several additional keys, which are used to control and edit the display.

c. Characters are displayed on the screen in a manner that resembles printed text. A typical full screen display is 24 rows by 80 columns (i.e., 1920 characters).

d. The display can normally be generated in two different modes:
   i. Scrolling mode in which lines appear at the bottom and move up the screen rather like credits on a movie screen.
   ii. Paging mode in which one complete screen-full is replaced by another, rather like a slide projector display.

e. Most VDUs have features that allow particular parts of the display to be highlighted or contrasted, e.g.,
   i. Inverse (Reverse) video, i.e., black on white instead of white on black.
   ii. Blinking displays.
   iii. Two levels of brightness.
   iv. Colour - on the more expensive models.

f. **Cursor** controls. A cursor is a small character-size symbol displayed on the screen, which can be moved about the screen both vertically and horizontally by means of special keys on the keyboard. During data input, the display may resemble a blank form. In this case data may be entered by first moving the cursor to a space on the "form" and then typing in the data. Further keys may allow the data to be edited or corrected.

g. Inbuilt microprocessors. The numerous internal functions of almost all modern VDUs are controlled by inbuilt microprocessors. The more expensive models are often called intelligent terminals.

How it works: When a key is pressed on the keyboard the character's code (in ASCII, say) is generated and transmitted to the computer along the lead connecting the terminal to the computer. Normally the character code received by the computer is immediately "echoed" back out to the terminal by the computer. When a character code is received by the terminal it is interpreted by the control circuitry and the appropriate character symbol is displayed on the screen, or printed depending upon the type of terminal. In what follows the basic operation of a VDU is covered in more detail.

For the more basic VDU models the character codes are stored in a memory array inside the VDU with each location in memory corresponding to one of the 24 x 80 character positions on the screen. The VDU is able to interpret control characters affecting the text format. Each character in the character set has its display symbol defined in terms of a grid of bits. These predetermined patterns are normally held in a special symbol table in ROM. Common grid sizes are 8 x 14 and 9 x 16. The character array is scanned by circuitry in the VDU and the character map generator refers to the ROM to produce the appropriate bit-map image for each character to be displayed on the screen. Often the device is able also to interpret sequences of control characters (often beginning with the ASCII control character) which may alter display - characteristics such as reverse video or colour. This kind of VDU is only able to

form images on the screen by constructing them from character symbols by means of the character map generator. It is therefore called a character terminal. The superior alternative is a graphics **terminal** which has high quality displays that can be used for line drawings, draughtsmen's drawings, etc. In a Raster Scan Display, which is just one of many types of display technology, the character codes received from the computer are interpreted by the terminal's map generator which then loads the appropriate hit pattern into special memory (video RAM) acting as a bit-map for the whole screen

Fig. 5a: Terminal

Workstations

Fig. 5b: Workstation

A typical workstation looks similar to a PC because it is a desktop computer with screen and keyboard attached. However, it is different in a number of respects as will be seen from the following list of essential features.

a. It is larger and more powerful than a typical PC.
b. It is fully connected into a computer network as another computer on the network in its own right and not just running a terminal emulator.
c. It has high-resolution graphics on bit-mapped screens as a standard feature. So it incorporates the capabilities of the graphics terminal.
d. It has a multi-tasking operating system which means that it is able to run multiple applications at the same time. This feature is now found in most present PCs.

Uses: Workstations are normally used by professionals for particular kinds of work such as Finance (dealer rooms), Science and Research, and Computer Aided Design. They are also very popular for programming.

Examples of workstations are the Digital VAXSTATIONs and the SUN SPARCstations.

Output Devices

Printers

The following devices and media will be described:

a. Printers - Single sheet or continuous stationery.
b. Microform recorder Microfilm or Microfiche.
c. Graph Plotters - Single sheet or continuous stationery.
d. Actuators.
e. Others.

Print Speeds tend to be expressed in terms of **cps** (characters per second), **lpm (**lines per minute) or **ppm** (pages per minute). Printers may be classified as:

a. **Low speed** (10 cps to approx. 300 lpm) - usually character printers.
b. **High speed** (Typically 300 lpm - 3000 lpm) - usually line printers or page printers.

Basic methods of producing print.

a. Impact or non-impact printing. Impact printers *hit* inked ribbons against paper whereas non-impact printers use other methods of printer, e.g., thermal or electrostatic. Most impact printers are noisy.

b. Shaped character or dot-matrix printing. A dot matrix can also be used to produce a whole picture or **image** similar in principle, but superior in quality, to the minute pattern of dots in a newspaper picture.

Low-speed printers

Dot matrix impact character printers

These were once the most popular and widely- used low-speed printers. They remain popular for general use where print quality is not critical and are widespread for special purposes. For example, small versions of these printers are often used in conjunction with computerised tills in shops and service stations, especially for dealing with purchases by credit card. They are often loosely referred to as **"dot matrix printers".**

Features

a. As with all character printers the device mimics the action of a typewriter by printing single characters at a time in lines across the stationery. The print is produced by a small **"print head"** that moves to and fro across the page stopping momentarily in each character position to strike a print ribbon against the stationery with an array of wires.
b. According to the number of wires in the print head, the character matrix may be 7 x 5, 7x 7, 9 x 7, 9 x 9 or even 24 x 24. The more dots the better the image.
c. Line widths are typically 80, 120, 132, or 160 characters across.
d. Speeds are typically from 30 cps to 200 cps.
e. Multiple print copies may be produced by the use of carboned paper (e.g. 4-6 copies using NCR (No Carbon Required) paper).

Some higher quality versions can produce NLQ(Near Letter Quality), have inbuilt alternative character sets plus features for producing graphs, pictures, and colour.

Inkjet printers

The original models of these printers were character matrix printers and had only limited success. Modern inkjet printers can act as character printers or page printers producing high print quality relatively quietly and have therefore replaced dot matrix printers for most low speed printing office use.

Fig 5c: HP DeskJet 850
Picture courtesy of Hewlett Packard

Features:

a. These are non-impact page printers often having inbuilt sets of scaleable fonts.

b. They operate by firing very tiny ink droplets onto the paper by using an "electrostatic field". By this means a medium quality bit-mapped image can be produced at a resolution of about 300-600dpi or above. Those using oil-based inks tend to produce higher quality print than those using water based inks.
c. They are very quiet but of low speed (4-6ppm). Their lower speed is reflected in the price.
d. Some models print colour images (typically at 2ppm), by means of multiple print heads each firing droplets of a different colour.
e. Some can print on plain paper, glossy paper and transparencies.

Daisywheel printers. This was once a popular type of low-speed printer that was favoured over dot matrix printers but is now far less common because it has been superseded by superior inkjet printers.

Features:

a. An impact shaped-character printer.
b. These printers are fitted with *exchangeable* print heads called daisywheels. To print each character the wheel is rotated and the appropriate spoke is struck against an inked ribbon.
c. Speeds are typically 45 cps.
d. They are similar to dot matrix printers in terms of page size and multiple-copy printing.

Other low-speed printers. There are many other low-speed printers too numerous to mention. One worth a mention is the Thermal printer which is a non-impact character matrix printer which prints onto special paper using a heated print head. It is very quiet and this gives it a big advantage for some applications.

Actuators

Computers are frequently used to control the operation of all manner of devices, appliances, mechanisms and processes. Any computer output device which is used to communicate some required action to a machine may be called an Actuator. For example a microcomputer operating a washing machine would use actuators to operate the motors and pumps. In a chemical plant controlled by computer actuators would operate valves, pumps, etc.

Other devices

It is becoming increasingly common for small loudspeakers to be fitted in desktop computers. Although this facility is often used for computer games there are a number of serious uses. One general use is as a means of providing messages to the user, the simplest form of which is a warning "bleep" sound when something is wrong. However, loudspeakers now come on

their own when used in conjunction with digitised sound. For example, by means of special software a desktop computer may be turned into a sound synthesiser unit which can be hooked up to an audio system.

Summary

The *features* of the *main* hardware units and media for the output of data from the computer have been covered. They are:

a. Printers - Single sheet or continuous stationery.
b. Microform recorder - Microfilm or Microfiche.
c. Graph Plotters - Single beet or continuous stationery.
d. Actuators.

Week Six

Concept of Data capture and data entry, problems of data entry, data collection stages, data capture techniques and devices, Computer file concepts, computer file processing, computer disk storage file processing and element of a computer file.

Objective:

- To explore data capturing that support data collections in a data processing.
- To discuss the general concept of data capture and data entry.
- To enable the students to implement appropriate data collection device for data processing.
- To introduce file concepts in computer followed by an extended discussion of the ways of view file store in computer and the purpose of data file in data processing environment and computer.

Description: data entry versus data entry, features of data capture devices and features of document captured from data capture device were explain to aim better understanding. To introduce file concepts in computer followed by an extended discussion of the ways of view file store in computer and the purpose of data file in data processing environment and computer

Data Capture and Data Entry

Introduction

These days *the majority of computer end-users input data to the computer :via keyboards on PCs, workstations or terminals.* However, for many medium and large scale commercial and industrial applications involving large volumes of data the use of keyboards is not practical or economical. Instead, specialist methods, devices and media are used and these are the subject of this segment. The segment begins by examining the problems of data entry. It goes on to consider the stages involved and the alternatives available. It then examines the factors that influence the choice of methods, devices and media for data input. Finally, the segment examines the overall controls that are needed over data as it is entered into the computer for processing. The selection of the best method of data entry is often the biggest single problem faced by those designing commercial or industrial computer systems, because of the high costs involved and numerous practical considerations. The best methods of data entry may still not give satisfactory facilities if the necessary controls over their use are not in place.

Data entry problems

The data to be processed by the computer must be presented in a machine-sensible form (ie, the language of the particular input device). Therein lies the basic problem since much data *originates* in a form that is *fear* from machine sensible. Thus a painful error-prone process of transcription must be undergone before the data is suitable for input to the computer.

The process of data collection involves getting the original data to the "processing centre", transcribing it, sometimes converting it from one medium to another, and finally getting it into the computer. This process involves a great many people, machines and much expense.

A number of advances have been made in recent years towards automating the data collection process so as to bypass or reduce the problems. This segment considers a variety of methods, including many that are of primary importance in commercial computing. Specialist methods used for industrial computer applications will be covered in later segments.

Data can originate in many forms, but the computer can only accept it in a **machine - sensible form.** The process involved in getting the data from its point of origin to the computer in a form suitable for processing is called **Data Collection.**

Before dealing with the individual stages of data collection it should be noted that data

collection *starts* at the source of the raw data and *ends* when valid data is within the computer in a form ready for processing.

Many of the problems of data entry can be avoided if the data can be obtained in a computer-sensible form at the point of origin. This is known as data capture. This segment will describe several methods of **data capture.** The capture of data does not necessarily mean its immediate input to the computer. The captured data may be stored in some intermediate form for later entry into the main computer in the required form. If data is input directly into the computer at its point of origin the data entry is said to be on-line. In addition, the method of direct input is a terminal or workstation method of input which is known as **Direct Data Entry (DDE).** The term **Data Entry** used in the segment title usually means not only the process of physical input by a device but also **any methods directly associated with the input**.

Data collection Stages

The process of data collection may involve any number of the following stages according to the methods used.

a. Data creation, e.g., on clerically prepared source documents.
b. Transmission of data.
c. Data preparation, i.e., transcription and verification.
d. Possible conversion from one medium (e.g., diskette) to another (e.g., magnetic tape
cartridge or magnetic disk).
e. Input of data to the computer for validation.
f. Sorting.
g. Control - all stages must be controlled.

Not all data will go through every stage and the sequence could vary in some applications. Even today, a high proportion of input data starts life in the form of a manually scribed or typewritten document and has to go through all the stages. However, efforts have been made to reduce the number of stages. Progress has been made in preparing the source document itself in a machine-sensible form so that it may be used as input to the computer without the need for transcription. In practice, the method and medium adopted will depend on factors such as cost, type of application, etc.

Character recognition

The methods described so far have been concerned with turning data into a machine sensible form as a prerequisite to input. By using Optical Character Recognition (OCR) and Magnetic Ink Character Recognition (MICR) techniques, the source documents *themselves* are prepared in a machine-sensible form and thus *eliminate* the transcription stage. Notice, however, that such characters can *also* be recognised by the human eye. We will first examine the devices used.

Document readers

Optical readers and documents. There are two basic methods of optical document reading:

a. Optical Character Recognition **(OCR).**

b. Optical Mark Recognition **(OMR).**

These two methods are often used in conjunction with one another, and have much in common. Their common and distinguishing features are covered in the next few paragraphs.

Features of an optical reader.

a. It has a document-feed hopper and several stackers, including a stacker for "rejected" documents.
b. Reading of documents prepared in optical characters or marks is accomplished as follows:
   i. **Characters**. A scanning device recognises each character by the amount of reflected light (i.e., OCR). The method of recognition, although essentially an electronic one, is similar in principle to matching photographic pictures with their negatives by holding the negative in front of the picture. The best match lets through the least light.
   ii. **Marks**. A mark in a particular position on the document will trigger off a response. It is the *position* of the mark that is converted to a value by the reader (i.e., OMR). The method involves directing thin beams of light onto the paper surface which are reflected into a light detector, unless the beam is absorbed by a dark pencil mark, i.e., a mark is recognised by the reduction of reflected light.

   Note. An older method of mark reading called mark sensing involved pencil marks conducting between two contacts and completing a circuit.
c. Documents may be read at up to 10,000 A4 documents per hour.

Features of a document.

a. Documents are printed in a stylised form (by printers, etc, fitted with a special typeface) that can be recognised by a machine. The stylised print is also recognisable to the human eye. Printing must be on specified areas on the document.
b. Some documents incorporate optical marks. Predetermined positions on the document are given values. A mark is made in a specific position using a pencil and is read by the reader.
c. Good-quality printing and paper are vital.
d. Documents require being undamaged for accurate reading.
e. Sizes of documents, and scanning area, may be limited.

Magnetic ink reader and documents

The method of reading these documents is known as Magnetic Ink Character Recognition (MICR).

Features of magnetic ink readers

a. Documents are passed through a strong magnetic field, causing the iron oxide in the ink encoded characters to become magnetised. Documents are then passed under a read head, where a current flows at a strength according to the size of the magnetised area (i.e., characters are recognised by a magnetic pattern).

b. Documents can be read at up to 2,400 per minute.

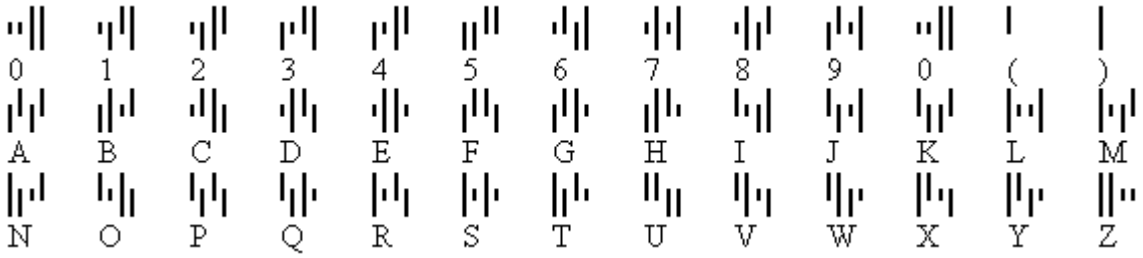


Fig. 6: OCR character specimen

Features of documents.

a. The quality of printing needs to be very high.
b. The characters are printed in a highly distinctive type style using ink containing particles of iron oxide, which gives the required magnetic property. Examine a bank cheque and security documents for a further example.

Optical character recognition (OCR)

a. Technique explained:

i. Alphabetic and numeric characters are created in a particular type style, which can be "read" by special machines. The characters look so nearly like "normal" print that they can *also* be read by humans.
ii. Characters are *created* by a variety of machines (e.g., line printers. typewriters, cash registers, etc) fitted with the special type face.
iii. The special optical character-reading machines *can* be linked to a computer in which the data is read from the document into the processor.

b. Applications:

OCR is used extensively in connection with billing, e.g., gas and electricity bills and insurance premium renewals and security printing. In these applications the bills are prepared in OC by the computer, then sent out to the customers, who return them with payment cheques. The documents re-enter the computer system (via the OC reader) as evidence of payment. This is an example of the "turnaround" technique. Notice that no transcription is required.

c. OCR/keyboard devices:

These permit a combination of OCR reading with manual keying. Printed data (e.g., account numbers) is read by OCR; hand-written data (e.g., amounts) is keyed by the operator. This method is used in credit card systems.

Optical mark reading (OMR)

a. Technique explained:

Mark reading is discussed here because it is often used in conjunction with OCR, although it must be pointed out that it is a technique in *itself*. Positions on a document are given certain values. These positions when "marked" with a pencil are interpreted by a machine. Notice it is the "position" that the machine interprets and that has a predetermined value.

b. Application:

Meter reader documents are a good example of the use of OMR in conjunction with OCR. The computer prints out the document for each customer (containing name, address, *last* reading, etc,) in OC. The meter reader records the current reading in the form of "marks" on the same document. The document reenters the computer system (via a reader that reads OC *and OM)* and is processed (i.e., results in a bill being sent to the customer). Note that this is another example of a "turnaround document".

Magnetic ink character recognition (MICR)

a. Techniques explained:

Numeric characters are created in a highly stylised type by special encoding machines using magnetic ink. Documents encoded thus are "read" by special machines.

b. Application. One major application is in banking (look at a cheque book), although some local authorities use for payment of rates by installments. Cheques are encoded at the bottom with account number, branch code and cheque number *before* being given to the customer (i.e., pre-encoded). When the cheques are received from the customers the bottom line is completed by encoding the *amount* of the cheque (i.e., post-encoded). Thus all the details necessary for processing are now encoded in MIC and the cheque enters the computer system via a magnetic ink character reader to be processed.

Data capture devices

The devices are mostly special-purpose devices intended for use in particular applications. Common, special and typical examples are described in the next few paragraphs.

Direct input Devices

a. Special sensing devices may be able to detect events as they happen and pass the appropriate data directly to the computer. For example:

i. On an automated production line, products or components can be "counted as they pass specific points. Error can stop the production line.

ii. At a supermarket checkout **a laser scanner** may read coded marks on food packets as the packets pass by on the conveyer. This data is used by the computerized till receipt and maintain records of stock levels (details later).

iii. In a computer-controlled chemical works, food factory or brewery industrial instruments connected to the computer can read temperatures and pressures in vats.

b. **Voice data entry (VDE) devices.** Data can be spoken into these devices. Currently they are limited to a few applications in which a small vocabulary is involved.

Features

The specific feature of these devices tends to depend upon the application for which they are used. However, data captured by the device must ultimately be represented in some binary from in order to processed by a *digital* computer. For some devices, the input may merely be a single bit representation that corresponds to some instrument, such as a pressure switch, being on or off.

Computer File Concepts

Files and file processing - part introduction

1. Files are named collections of stored data. Files tend to be too large to be held in main storage and are therefore held on backing storage devices such as magnetic disks. When data in a file is required for processing the file is read into main storage in manageable amounts.

2. Named programs are also often held on backing storage ready for use and may be regarded as "files" too. In this Part we are primarily concerned with files stored for data processing which we may call "data files" to distinguish them from files containing programs. The term "data file" is interpreted broadly in this context to include text files or document files.

3. Traditionally individual programs were written to process individual files or groups of files and this is still the case for many simple or specialist applications. However, files are also used as the building blocks for databases as will be described in later segments. As we will see, the database approach to the processing of data has some significant advantages over file processing in many situations but to understand why it is first

necessary to have a basic grounding on how file processing works. Also, most of the basic file processing methods are still used within database systems although their use may not be evident to a user. This means that file processing is important both in its own right and as an underlying feature of database processing.

5. Others discuses the concepts of magnetic files and describes the methods of organising a file on *disk* (the main on-line storage medium for files) and how access is made to the records in such a file.
6. In the interests of clarity only one storage device (disk) is used as a basis for discussing computer files. More generally, the devices for file storage can be classified as follows:
    a. Disk (Magnetic hard or floppy, plus optical) - Direct Access Storage (DAS). Used for on-line file processing.
    b. Magnetic tape - Serial Access Storage (SAS). Once used extensively for on-line file processing but now mostly used to store files off-line e.g. for backup or archiving.
7. The *general* principles discussed with regard to *disk* can be applied to other SAS media.

Introduction

The purpose of this segment is to look at the general concepts that lie behind the subject of computer files before going on to discuss the different methods of organising them. At all times the term "file" will refer to computer data files.

Purpose data file

A file holds data that is required for providing information. Some files are processed at regular intervals to provide this information (e.g., payroll file) and others will hold data that is required at regular intervals (e.g., a file containing prices of items).

There are two common ways of viewing files:

a. Logical files. A "logical file" is a file viewed in terms *of what* data items its records contain and *what* processing operations may be performed upon the file. The user of the file will normally adopt such a view.

b. Physical files. A "physical file" is a file viewed in terms of *how* the data is stored on a storage device such as a magnetic disk and *how* the processing operations are made possible

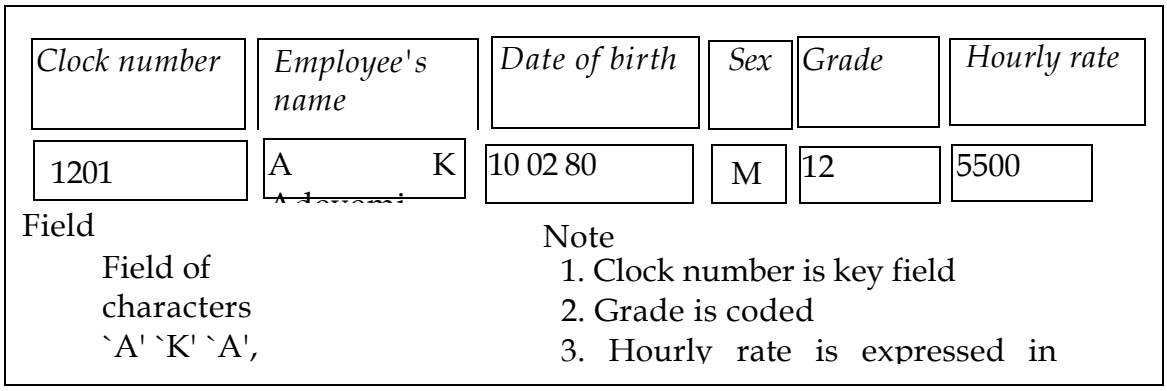


Fig 7a: Payroll Record System

A logical file can usually give rise to a number of alternative implementations. These alternatives are considered in later segments.

Elements of a computer file

A file consists of a number of records. Records were defined earlier. Here we consider records in a slightly different way because we are concerned with the way they are commonly stored. Each record is made up of a number of **fields** and each field consists of a number of **characters.**

a. Character. A character is the smallest element in a file and can be alphabetic, numeric or special.

b. Field. An item of data within *a record* is called a field - it is made up of a number of *characters,* e.g., a name, a date, or an amount.

c. Record. A record is made up of a number of related fields, e.g., a customer record, or an employee payroll record (see Fig. 7a).

Alternative terminology

The terminologies of record, field and character are firmly established as a means of describing the characteristics of files in general situations. However, the use of this terminology can lead to excessive attention being directed towards physical details, such as how many characters there should be in a field. Such issues can divert attention from matters of high priority, such as what fields should be recorded in order to meet the information needs of the user. To overcome this difficulty, two alternative bets of terms have been developed, one set for physical files, and the other set for logical files. They are:

a. For physical files.
   i. Physical record.
   ii. Field.
   iii. Character (a physical feature).

b. For logical files.
   i. Logical record - an "entity".
   ii. Data item - "attributes" of the "entity".

Entities are things (e.g., objects, people, events, etc.) about which there is a need to record data, e.g., an item of stock, an employee, a financial transaction, etc. The individual properties of the entity, about which data is recorded, are its "attributes", e.g., the attributes of an invoice (entity) will include the "name"; "address"; "customer order number"; "quantity"; "price"; "description".

A logical record is created for each entity occurrence and the logical record contains one data item for each occurrence of the entity's attributes, e.g., the "customer's name" would be a data item and there would be one only in the logical record whereas the attribute "quantity" would have as many data items as there are entries on the invoice. The relationship between the various terms used is summarised in Fig. 7b. (opposite)

Things about which there is a need to record data	Entities	• each entity has a number of *Attributes*
How the data is recorded	Logical records (1 per entity occurrence)	• each logical record contains a number of *data items*
Physical details of how the data is recorded	Physical record (1 or more per logical record)	• each physical record contains a number of

Fig. 7b: Entities and Attributes.

Week Seven

Mid Semester Test

Objective:

- To evaluate performance of student knowledge on the Lectures/teachings received this Course

Description: To know how far the student can apply the knowledge gain from the course.

Week Eight and Nine

Practical work with SQL or MS Access or SPSS

Objective:

- To teach the Practical aspect of data processing.
- To implement some of discussed techniques used in this course

Description: Demonstrate of the fields, records, files including database are created using query language approach.

INTRODUCTION

Database contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data. Below is an example of a table called "Persons":

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Table 8.0: Table Persons

The file table above contains three records (one for each person) and five columns (P_Id, LastName, FirstName, Address, and City

Using SQL, you can create the table structures within the database you have designated. For example, the STORAGE table would created with:

CREATE TABLE STORAGE(LOC_ID CHAR(12) NOT NULL, ITEM_ID CHAR(10) NOT NULL, STOR_QTY NUMBER, PRIMARY KEY(LOC_ID,ITEM_ID), FOREIGN KEY(LOC_ID) REFERENCES LOCATION ON DELETE RESTRICT ON UPDATE RESTRICT, FOREIGN KEY(ITEM_ID) REFERENCES ITEM ON DELETE CASCADE ON UPDATE CASCADE);

Most DBMSs now use interfaces that allow you to type the attribute names into a template and to select the attribute characteristics you want from pick lists. You can even insert comments that will be reproduced on the screen to prompt the user for input. For example, the preceding STORAGE table structure might be created in a

u want to generate a LA schedule , you need data from two tables. LABASSISTANT and WORK-SCHEDULE. Because the report output is ordered by semester, LA, weekday, and time, indexes must be available for the primary key fields in each table. Using SQL, we would type:

CREATE UNIQUE INDEX LAS_DEX ON LAB_ASSISTANT(LA_ID) and CREATE UNIQUE INDEX WS_DEX ON WORK SCHEDULE(SCHED_SEMESTER, LA_ID, SCHED_WEEKDAY, SCHED_TIME_IN);

Most modern DBMSs automatically index on the primary key components. Views are often for security purposes. However, views are also used to streamline the system's processing requirements. For example, output limits may be defined efficiently appropriate views necessary for the LA schedule report for the fall semester of 1999, we use the CREAT VIEW command:

CREATE VIEW LA_SCHED AS SELECT LA_ID, LA_NAME, SCHED_WEEKDAY, SCHED_TIME_IN SCHED_TIME_OUT WHERE SCHED_SEMESTER = 'FALL99';
The designer creates the view necessary for each database output operation.

USING SQL ON A RELATIONAL DATABASE

SQL can be used on MySQL, Oracle, Sybase, IBM DB2, IBM Informix, Borland Interbase, MS Access, or any other relational database system. This unit uses MySQL to demonstrate SQL and uses MySQL, Access, and Oracle to demonstrate JDBC programming. Assume that you have installed MySQL with the default configuration, you can access MySQL from the DOS command prompt using command MySQL from c./MySQL/bin directory, as shown in figure below.

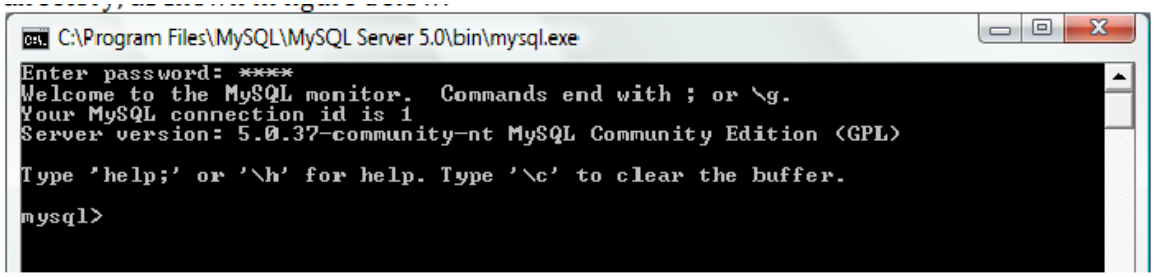
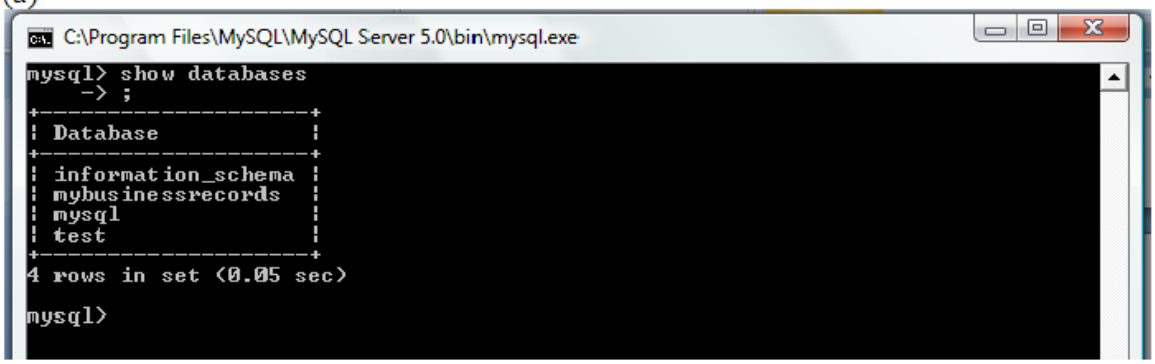


Figure 8.1 You can access a MySQL database server from the command window.

(a)



(b)

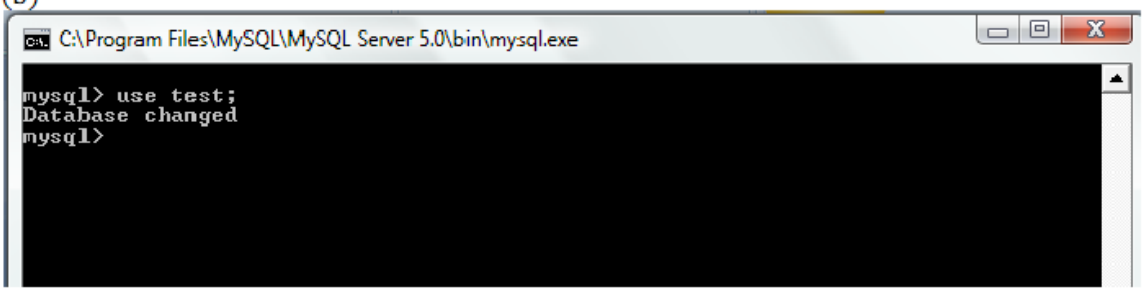


Figure 8.2 (a) The show database command display all available databases in the MySQL
database server; (b) The use test command selects the test database. The MySQL database contains the tables that store information about the server and its users. This database is intended for the server administrator to use. For example, the administrator can use it to create users and grant or revoke user privileges. Since you care the owner of the server installed on your system, you have full access to the MySQL database. However, you should not create user tables in the MySQL database. You can use the test database to store data or create new databases. You can also create a new database using the command create database <database name> or drop an existing database using the command drop database <database name>. To select a database for use, type use database command. Since the test database is created by default in every MySQL database, let use it to demonstrate SQL commands. As shown in the figure above, the test database is selected. Enter the statement to create the course table as shown in figure below:

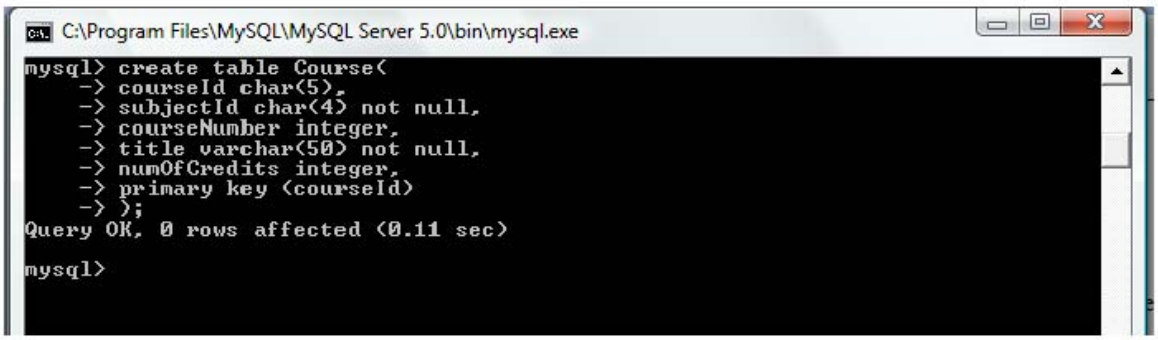


Figure 8.3 The execution result of the SQL statements is displayed in the MSQL monitor

If you make typing errors, you have to retype the whole command. To avoid retyping the whole command, you can save the command in a file, and then run the command from the file. To do so, create a text file to contain the commands, named, for example, test.sql. You can create the text file using any text editor, such as notepad, as shown in the figure below. To comment a line, proceed it with two dashes. You can now run the script file by typing source test.sql from MySQL command prompt, as shown in the figure below

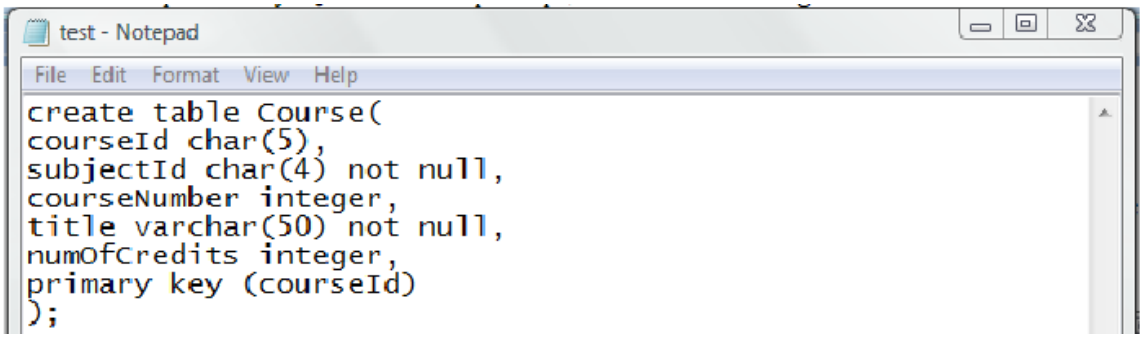


Figure 11.4 You can use Notepad to create a text file for SQL commands

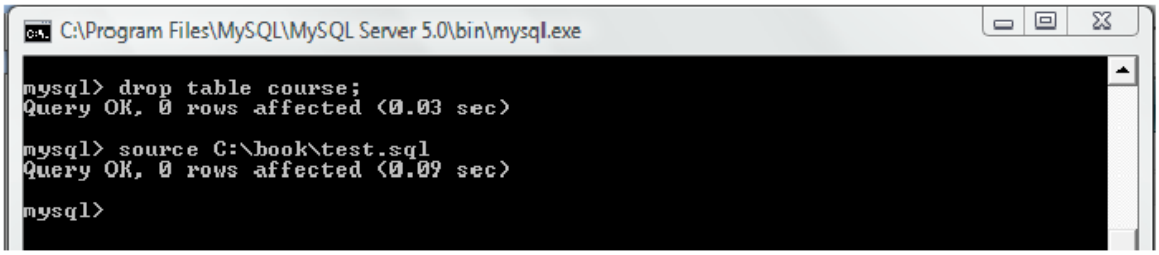


Figure 8.5: You can run the SQL commands in a script file from MySQL

SQL STATEMENTS

The table below contains a list of SQL commands and functions:

SQL Basic	SQL Advanced	SQL Functions
SQL Syntax	SQL Top	SQL avg()
SQL Select	SQL Like	SQL count()
SQL Distinct	SQL Wildcards	SQL first()
SQL Where	SQL In	SQL last()
SQL And & Or	SQL Between	SQL max()
SQL Order By	SQL Alias	SQL min()
SQL Insert	**SQL Joins**	SQL sum()
SQL Update	SQL Inner Join	SQL Group By
SQL Delete	SQL Left Join	SQL Having
	SQL Right Join	SQL ucase()
	SQL Full Join	SQL lcase()
	SQL Union	SQL mid()
	SQL Select Into	SQL len()
	SQL Create DB	SQL round()
	SQL Create Table	SQL now()
	SQL Constraints	SQL format()
	SQL Not Null	SQL

Table 8.2: SQL Commands and Functions SQL Basic SQL Advanced SQL Func

CREATING AND DROPPING TABLES
Is null()

Tables are the essential objects in a database. To creates a table, use the create table statement to specify a table name, attributes, and types, as in the following example
create table Course(courseId char(5), subjectId char(4) not null, courseNumber integer, title varchar(50) not null, numOfCredits integer, primary key (courseId));
This statement creates the course table with attributes courseId, subjectId, courseNumber, title and numOfCredits. Each attribute has a data type that specifies the type of data stored in the attribute. char(5) specifies that courseId consists of five characters. varchar(50) specifies that title is a variant-length string with a maximum of fifty characters. Integer specifies that courseNumber is an integer. The primary key is courseId. The table Student and Enrollment can be created as follows:
create table Student (ssn char(9) firstName varchar(5), mi char (1) lastName varchar (25) birthDate date, street varchar (25), phone char(11) zipCode char (5), deptId char(4), primary key (ssn)); create table Enrollment (ssn char(9), courseId char(5) dateRegistered date, grade char (10, primary key (ssn, courseId) foreign key (ssn) references student, foreign key (courseId) references Course);

If a table is no longer needed, it can be dropped permanently using the drop table command. For example, the following statements drops the Course table:
drop table Course; If a table to be dropped is referenced by other tables, you have to drop other tables first. For example, if you have created the tables Course, Student and Enrollment and want to drop Course, you have to first drop Enrollment, because Course is referenced by Enrollment.

THE SQL SELECT STATEMENT
The SELECT statement is used to select data from a database. Depending on the form it takes, it searches through the table in the database and selects the data that matches the criteria. The result is stored in a result table, called the result-set.

SQL SELECT Syntax SELECT column_name(s) FROM table_name and SELECT * FROM table_name

Note: SQL is not case sensitive. SELECT is the same as select. An SQL SELECT Example The "Persons" table:

P_Id	LastName	FirstName	Address		City
1	Hansen	Ola	Timoteivn 10		Sandnes
2	Svendson	Tove	Borgvn 23		Sandnes
3	Pettersen	Kari	Storgt 20		Stavanger

Table 8.2: The "Persons" table

Now we want to select the content of the columns named "LastName" and "FirstName" from the table above. We use the following SELECT statement: SELECT LastName, FirstName FROM Persons. The result-set will look like this:

LastName	FirstName
Hansen	Ola
Svendson	Tove
Pettersen	Kari

SELECT * Example Now we want to select all the columns from the "Persons" table. We use the following SELECT statement: SELECT * FROM Persons

Tip: The asterisk (*) is a quick way of selecting all columns! The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

The SQL SELECT DISTINCT Statement

In a table, some of the columns may contain duplicate values. This is not a problem,however, sometimes you will want to list only the different (distinct) values in a table. The DISTINCT keyword can be used to return only distinct (different) values. SQL SELECT DISTINCT Syntax

SELECT DISTINCT column_name(s) FROM table_name

SELECT DISTINCT Example Now we want to select only the distinct values from the column named "City" from the

table above. We use the following SELECT statement: SELECT DISTINCT City FROM Persons The result-set will look like this:

City
Sandnes
Stavanger

The WHERE clause is used to filter records.

The WHERE Clause The WHERE clause is used to extract only those records that fulfill a specified criterion. SQL WHERE Syntax

SELECTcolumn_name(s)

The WHERE clause is used to extract only those records that fulfill a specified criterion. SQL WHERE Syntax SELECTcolumn_name(s) FROMtable_name WHERE column_name operator value WHERE Clause Example The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select only the persons living in the city "Sandnes" from the table above. We use the following :
SELECT * FROM Persons WHERE City='Sandnes'
The result-set will look like this:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

Quotes Around Text Fields
SQL uses single quotes around text values (most database systems will also accept double quotes). Although, numeric values should not be enclosed in quotes. For text values: This is correct:
SELECT * FROM Persons WHERE FirstName='Tove'
This is wrong:
SELECT * FROM Persons WHERE FirstName=Tove
For numeric values: This is correct:
SELECT * FROM Persons WHERE Year='1965'
This is wrong:
SELECT * FROM Persons WHERE Year=1965
Operators Allowed in the WHERE Clause
With the WHERE clause, the following operators can be used:

Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	If you know the exact value you want to return for at least one of the columns

Note: In some versions of SQL the <> operator may be written as != The AND & OR operators are used to filter records based on more than one condition. 3.4.7. The AND & OR Operators The AND operator displays a record if both the first condition and the second condition is true. The OR operator displays a record if either the first condition or the second condition is true. AND Operator Example The "Persons" table:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

Now we want to select only the persons with the first name equal to "Tove" AND the last name equal to "Svendson": We use the following SELECT statement:

```
SELECT * FROM Persons
WHERE FirstName='Tove'
AND LastName='Svendson'
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

OR OPERATOR EXAMPLE

Now we want to select only the persons with the first name equal to "Tove" OR the first name equal to "Ola":
We use the following SELECT statement:

```
SELECT * FROM Persons
WHERE FirstName='Tove'
OR FirstName='Ola'
```

The result-set will look like this:
P_Id LastName FirstN1 Hansen Ola 2 Svendson Tove

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes

COMBINING AND & OR

You can also combine AND and OR (use parenthesis to form complex expressions).

Now we want to select only the persons with the last name equal to "Svendson" AND the first name equal to "Tove" OR to "Ola":
We use the following SELECT statement:

```
SELECT * FROM Persons WHERE
LastName='Svendson'
AND (FirstName='Tove' OR FirstName='Ola')
```

The result-set will look like this:

P_Id	LastName	FirstName	Address	City
2	Svendson	Tove	Borgvn 23	Sandnes

Week Ten

Types of file, access to file, Storage devices, Processing activities of files, Fixed-length and variable-length records, Hit rate

Objectives:

- To discuss the general concept of types of file and access to file in data processing.
- Impact to student the knowledge of processing file in computing environment.
- To discuss the processing activities of files and it application to computer file.
- To have know how required in choosing between Fixed-length and variable-length records in record design.

Description: updating master file, Transaction File, Reference file interrogation, file characteristics, major processing activities are discussed in detail and features were emphasized

Types of files

a. **Master file.**
These are files of a fairly permanent nature, e.g., customer ledger, payroll, inventory, etc. A feature to note is the regular *updating* of these files to show a current position. For example customer's orders will be processed, increasing the "balance owing" figure on a customer ledger record. Therefore the master records will contain both data of a static nature, e.g., a customer name and address, and data that, by its nature will change each time a transaction occurs, e.g., the "balance" figure already mentioned.

b. **Transaction File.**
Also called Movement file. This is made up of the various transactions created from the source documents. In a sales ledger application the file will contain all the orders received at a particular time. This file will be used to update the *master file.* As soon as it has been used for this purpose it is no longer required. It will therefore have a very short life, because it will be replaced by a file containing the *next* batch of orders.

c. **Reference file.**
A file with a reasonable amount of permanency. Examples of data used for reference purposes are price lists, tables of rates of pay, names and addresses, student transcript.

Access to files

Key fields: When files of data are created one needs a means of access to particular records within those files. In *general* terms this is usually done by giving each record a "key" field by which the record will be recognised or identified. Such a key is normally *a unique identifier* of a record and is then called the **primary** key. Sometimes the primary key is made from the combination of two fields in which case it may be called a *composite key or compound key*. Any other field used for the purpose of identifying records, or sets of records, is called a secondary key. Examples of primary key fields are:

a. Customer number in a customer ledger record.
b. Stock code number in a stock record.
c. Employee clock number in a payroll record.

Not only does the key field assist in accessing records but also the records themselves can, if required, be sorted into the sequence indicated by the key.

Storage devices

The two storage devices that may be considered in connection with the storage of files (i.e., physical files).

a. Magnetic or optical disk. These are direct access media and are the *primary means of storing files on-line*
b. Magnetic tape. This medium has significant limitations because it is a serial access medium and therefore is the *primary means of storing files off-line.*

These characteristics loom large in our considerations about files in the segments that follow. Note then that they are inherent in the *physical* make-up of the devices and will clearly influence what *types* of files can be stored on each one, and how the files can be *organised* and accessed.

Processing activities

We will need to have access to particular records in the files in order to process them. The major processing activities are given below:

a. **Updating.** When data on a master record is changed to reflect a current position, e.g., updating a customer ledger record with new orders. Note that the old data on the record is replaced by the new data.
b. **Referencing.** When access is made to a particular record to ascertain what is contained therein, e.g., reference is made to a "prices" file during an invoicing *run* Note that it does *not* involve any alterations to the record itself.
c. **File maintenance.** New records must be added to a file and records need to be deleted. Prices change, and the file must be altered. Customers' addresses also change and new addresses have to be inserted to bring the file up to date. These particular activities come under the heading of "maintaining" the file. File maintenance can be carried out as a separate run, but the insertions and deletions of records are sometimes *combined* with updating.
d. **File enquiry or** interrogation. This is similar in concept to referencing, it involves the need to ascertain a piece of information from, say, a master record. For example, a customer may query a statement sent to him. A "file enquiry" will get the data in dispute from the record so that the query may be settled.

Fixed-length and variable-length records

The question whether to use records of a fixed or variable length is one that usually does not have to be considered in manual systems.

a. **Fixed.** Every record in the file will be of the same fixed number of fields and characters and will never vary in size.
b. **Variable.** This means that not *all* records in the file will be of the same size. This could be for two reasons:

i. Some records could have more *fields* than others. In an invoicing application, for example (assuming a 6-character field to represent "total amount for each invoice"), we would add a new field to a customer record for each invoice. So a customer's record would vary in *size* according to the *number* of invoices he had been sent.
ii. Fields *themselves* could vary in size. A simple example is the "name and address" field because it varies widely in size.

It should be noted, however, that in the examples a fixed-length record *could* be used. The record could be designed in the first instance to accommodate a fixed number of *possible* invoices. This means that records with less than the fixed number of invoices would contain blank fields. Similarly in the figure above, the field could be made large enough to accommodate the *largest* name and address. Again records with names and addresses of a smaller number of characters would contain blanks. Fixed-length records make it easy for the programmer because he or she is dealing with a known quantity of characters each time. On the other hand they result in less efficient utilisation of storage. Variable-length records mean difficulties for the programmer but better utilisation.

Hit rate

This is the term used to describe the rate of processing of master files in terms of active records. For example, if 1,000 transactions are processed each day against a master file of 10,000 records, then the hit rate is said to be 10%. Hit rate is a measure of the "activity" of the file.

Other file characteristics

Apart from activity, which is measured by hit rate, there are other characteristics of the file that need to be considered. These are:

i. **Volatility.** This is the frequency with which records are added to the file or deleted from it. If the frequency is high, the file is said to be volatile. A file that is not altered is "static". If the frequency is low, the file is said to be **"semi-static".**
ii. **Size.** This is the amount of data stored in the file. It may be expressed in terms of the number of characters or number of records.
iii. **Growth.** Files often grow steadily in size as new records are added. Growth must be allowed for when planning how to store a file.

Note:

a. Magnetic tape is a serial –access medium, disk is a direct- access medium. Nevertheless disk can act as a serial –access medium if required.
b. **Direct access** should be distinguished from **immediate access** which refers to access to main store.
c. Direct access is also called random access because it means access in no set order.
d. In terms of levels of storage direct access is below immediate access and above serial access.

e. A file may be described in terms of its "structure" and in terms of its organisation. Its *structure* is determined by which data items are included in records and how the data items are grouped within records. Its *organisation* is determined by how the records are arranged within the file.
f. Files are not normally held in primary storage (i.e., main storage). They are normally held on an on-line backing storage (secondary storage) or on off-hiine hacking storage.
h. Files that are only processed occasionally are normally held on of line backing storage.

Study questions:

1. An organisation runs a simple savings scheme for its members. Members pay in sums of money to their own accounts and gain interest on the money saved. Data about the accounts is stored in a master file. What would you suggest would be the entities used in this system, Also suggest what attributes these entities might have.
2. Define the term "key field". Discuss the suitability of the following data items as key fields.
   a. A person's surname in a personnel file.
   b. A national insurance number in a payroll file.
   c. A candidate number in an examinations file.
3. Define the terms "hit rate" and "volatility" with regard to computer files. Where else have you come across the term "volatility" in computer science.

File Organisation and Access

Introduction

This segment describes the ways in which files may be organised and accessed on disks. Before tackling this segment the need to be thoroughly conversant with the relevant physical attributes of disks (fixed and exchangeable) and disk units ("reading" and "writing" devices).

in a master file is required. Comment on the probable characteristics of the file

a. Volatility,
b. Activity,
c. Size,
d. Growth.

Today most file processing is carried out using files stored on hard magnetic disks. Optical disks only have a minority use at present although they are being used increasingly for applications requiring large volumes of archived or reference data. Flash and Floppy disks are not normally used as file processing media because of their limited capacity. They are more often used to transfer small files between computers, particularly PCs. They are only used as the main file processing medium on a few very small microcomputers. The principles covered by this segment concerning the use of disks are applicable to all disk types. Any relevant differences will be highlighted when appropriate.

There is still some file processing carried out using files stored on magnetic tape but it is almost all done on mainframes in large commercial, industrial or financial institutions. Magnetic tape continues to be an important backup medium especially in its cartridge forms.

The simplest methods of organising and accessing files on disk are very similar to the standard ones used for magnetic tape. Where appropriate this similarity will be drawn to the reader's attention. Otherwise little mention will be made of magnetic tape.

File organisation is the arrangement of records within a particular file. We start from the point where the individual physical record layout has been already designed, i.e., the file "structure" has already been decided. How do we organise our many hundreds, or even thousands, of such records (e.g., customer records) on disk? When we wish to access one or more of the records how do we do it? This segment explains how these things are done.

Writing on disk

In order to process files stored on disk the disk cartridge pack must first be loaded into a disk unit. For a fixed disk the disk is permanently in the disk unit. Records are "written" onto a disk as the disk pack revolves at a constant speed within its disk unit. Each record is written in response to a "write" instruction, Data goes from main storage through a read-write head onto a track on the disk surface. Records are recorded one after the other on each track. (On magnetic tape the records are also written one after the other along the tape.)

> **Note.** All references to "records" in this segment should be taken to mean "physical records" unless otherwise stated.

Reading from disk

In order to process files stored on disk the disk cartridge or pack must first be loaded into a disk unit. Records are read from the disk as it revolves at a constant speed. Each record is read in response to a "read" instruction. Data goes from the disk to the main storage through the read-write head already mentioned. Both reading and writing of data are accomplished at a fixed number (thousands) of bytes per second.

W e will take for our discussion on file organisation a "6-disk" pack, meaning it has ten usable surfaces (the outer two are not used for recording purposes). But before describing how files are organised let us look first at the basic underlying concepts.

Cylinder concept

Where the disk pack is illustrated, and note the following:

i. There are *ten* recording surfaces. Each surface has 200 tracks.
ii. There is a read-write head for *each* surface on the disk pack.
iii. *All* the read-write arms are fixed to *one* mechanism and are like a comb.
iv. When the "access" mechanism moves all ten read-write heads move *in unison* across the disk surfaces.
v. Whenever the access mechanism comes to rest *each* read-write head will be positioned on the equivalent track on *each* of the ten surfaces.
vi. For *one* movement of the access mechanism access is possible to

ten tracks of data.

In the case of a floppy disk the situation is essentially the same but simpler. There is just *one* recording surface on a "single-sided" floppy disk and *two* recording surfaces on a "double-sided" floppy disk. The other significant differences are in terms of capacity and speed.

Uses made of the physical features already described when organising the storage of records on disk. Records are written onto the disk starting with track 1 on surface 1, then track 1 on surface 2, then track 1 on surface 3 and so on to track 1 on surface 10. One can see that conceptually the ten tracks of data can be regarded as forming a **CYLINDER**.

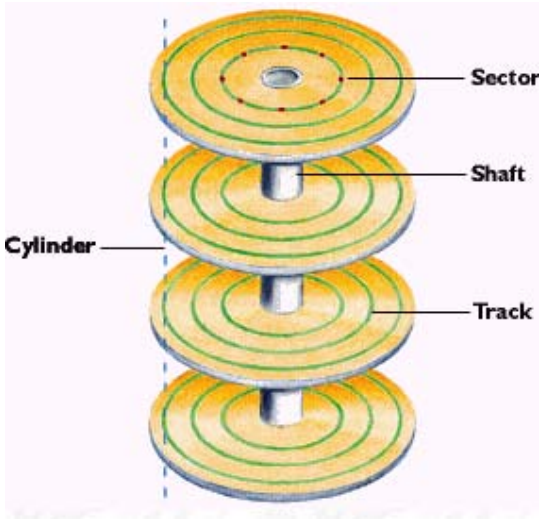


Fig. 8: Disk Cylinder

Data is written onto successive cylinders, involving *one* movement only of the access mechanism for each cylinder. When access is made to the stored records it will be advantageous, in terms of keeping access mechanism movement to a minimum, to deal with a cylinder of records at a time.

Conceptually the disk can be regarded as consisting of 200 CYLINDERS. Cylinder 1 comprises track 1 on each of 10 surfaces; cylinder 2 comprises track 2 on each of the 10 surfaces and so on to cylinder 200, which comprises track 200 on each of the 10 surfaces. This CYLINDER CONCEPT is fundamental to an understanding of how records are organised on disks. An alternative term for cylinder is SEEK AREA, i.e., the amount of data that is available to the read-write heads as a result of one movement or SEEK of the access mechanism.

Hard-sectored disks and soft-sectored disks

The tracks on a disk are subdivided into sector. There are two alternative design strategies for the division of tracks into sectors. One is called soft sectoring, the other is called hard sectoring.
In either case *whole* sectors of data are transferred between the disk and main storage. Note. A disk said to have "no sectors" is effectively a soft-sectored disk.

A soft-sectored disk has sectors that may be varied in length, up to some maximum value that is never more than the size of a complete track. Sector size and position is *software* controlled, hence the term "soft sectored".

A hard-sectored disk has sectors of fixed length. There may be anything from 8-128 sectors per track. Sector size and position is pre-determined by hardware, hence the term "hard sectored".

You may compare a sector with a physical record or block on magnetic tape. In fact, it is common to use "sector" and "block" as synonyms. However, in the case of hard-sectored disks, blocks may be grouped together into larger units called "buckets" or "logical blocks". It is therefore prudent to call a sector a "physical block" so as to avoid any possible ambiguity.

Week Eleven

Basic address concepts of disk files, Access time, File organisation on disk, Access , Methods of addressing, File labels, Control totals, Buffers and buffering

Objective:

- To introduce students to storage address creation, arrangement of stored file on storage media and access in different data processing environment as the fundamental building blocks of knowing the relevant of computer locating stored file.
- To explore the storage address concepts including fetching on the storage media.

Description: the organisation, access time and method of addressing of Serial Sequential file organisation, Index Sequential Organisation, Random file organisation were discuss in detail.

Basic address concepts

As the disk is a direct-access device a record can be accessed independently of other records. To enable the disk to locate a record, the record must have some form of ADDRESS. The whole area of each disk can be subdivided to enable this to be accomplished.

a. *Cylinder*: The major subdivision as we have seen is the cylinder.
b. *Track*: Each cylinder is composed of a number of tracks (10 in our quoted example).
c. *Block*: The smallest addressable part of a disk is a block (i.e., a sector). This forms the **unit** of transfer between the disk and main storage.
d. *Bucket*: When the block size is fixed (i.e., when the disk is hard sectored) a number of blocks (i.e., sectors) are grouped to form a larger unit of transfer. This unit is called a bucket, but may also be called a cluster, allocation unit or logical block.

The block or the bucket is therefore the unit of input/output on disk. (The same is true for magnetic tape where records are combined together into blocks that are treated as single units for purposes of reading or writing.) The optimum size for a block or bucket is determined by:

a. The need to optimise Pillage of a track - e.g., if a size were chosen for a variable length block that did not divide exactly into the track size, space would be wasted.

b. The need to minimise the number of transfers between main storage and disk storage. The larger the block or bucket size, the more records will be brought into main storage at each transfer.
c. The need to economise in the use of main storage. A large block or bucket may occupy so much main storage that insufficient space is left for other data and programs. Note. The number of records comprising a block is known as the blocking factor.

In basic hardware terms, the address of a record is given thus:

a. cylinder number
b. track number
c. block number (which will be the first block in the bucket if this concept applies) Note. On a single-sided floppy disk the address would be simply track number and block number.
Thus the address 1750906 indicates
i. cylinder 175
ii. track 09
ii. block 06

The start of each track is marked physically by a notch or hole. The start of each sector is marked by special data recorded the start of each track and sector is marked physically by a notch or hole. There may also be recorded marks as on the soft-sectored disk.

b. Hard-sectored disk

Key
Sectors (i.e. blocks) are numbered 1, 2, 3, ... Logical records
are numbered R1, R2, R3, ... indicates wasted storage space.

Access time
Access time on *disk is* the time interval between the moment the command is given to transfer data from disk to main storage and the moment this transfer is completed. It is made up of three components:

a. Seek time. This is the time it takes the access mechanism to position itself at the appropriate cylinder.

b. Rotational delay. This is the time taken for the bucket to come round and
position itself under the read-write head. On average this will be the time taken for
half a revolution of the disk pack. This average is called the "latency" of the disk.

c. Data transfer time. This is the total time taken to *read* the contents of the bucket
into main storage.

Access time will vary mainly according to the position of the *access mechanism* at the *time* the command is given. For example if the access mechanism is already positioned at cylinder 1 and the record required happens to be in cylinder 1 no movement of the access mechanism is required. If, however, the record required is in

cylinder 200, the access mechanism has to move right across the surface of the disk. Once the bucket has arrived at the read-write head, the transfer of data to storage begins. *Speed of transfer* of data to main storage is very fast and is a constant rate of so many thousand bytes per second. A hard disk will operate at speeds roughly 10 times faster than a floppy disk and flash disk.

File organisation on disk

There are four basic methods of organising files on disk:

a. Serial. Records are placed onto the disk one after the other with no regard for sequence. (This method can be used with magnetic tape.)
b. Sequential. Records are written onto the disk but in a defined sequence according to the record keys. (Again this method can be used with magnetic tape.)
c. Indexed sequential. Records are stored in sequence but with one important difference - an *index* is provided to enable individual records to be located. Strictly speaking the records may not always be stored in sequence but the index will always enable the sequence to be determined.
d. Random. Records are actually placed onto the disk "at random", that is to say there is no *obvious* relationship between the records. A mathematical formula is derived which, when applied to each record key, generates an answer, a bucket address. The record is then placed onto the disk at this address, e.g., one possible formula might be: given key = 33871, divide by 193 giving 175 remainder 96. (Address is taken as cylinder 175, track 9, block 6.)

Access

a. Serial files. The only way to access a serially organised file is SERIALLY. This simply means to say that each record is read from the disk into main storage one after the other in the order they occur on the disk. (This method can be used with magnetic tape).
b. Sequential **files.** The method of access used is still SERIAL but of course the file is now in sequence, and for this reason the term SEQUENTIAL is often used in describing serial access. It is important to note that to process (e.g., update) a sequential master file, the transaction file must *also* be in the sequence of the master file. Access is achieved by first reading the transaction file and then reading the master file until the matching record (using the record keys) is found. Note therefore that if the record required is the twentieth record on the file, in order to get it into storage to process it the computer will first have to read in *all* nineteen preceding records. (This method can be used with magnetic tape).

Note. Magnetic tape is limited to methods (a) and (b) above. These limited methods of organisation and access have led to tape becoming very much less common than disk as an on-line medium for the storage of master files. Tape continues as a major storage **medium for purposes such** as offline data storage and back-up.

c. Indexed sequential **files.** There are three methods of access:
   i. **Sequential.** This is almost the same as in (b) above; the complete file is read in

sequential order using the index. The method is used when the hit rate is high. The method makes minimal use of the index, minimises head movement and processes *all* records in each block in a single read. Therefore, the index is used once per block rather than once per record. Any transaction file must be pre-sorted into the same key sequence as the master file.

ii. **Selective sequential.** Again the transaction file must be pre-sorted into the same sequence as the master file. The transaction file is processed against the master file and *only* those master records for which there is a transaction are selected. Notice that the access mechanism is going forward in an ordered progression (never backtracking) because both files are in the same sequence. This minimises head movement and saves processing time. This method is suitable when the hit rate is low, as only those records for which there is a transaction are accessed.

iii. **Random.** Transactions are processed in a sequence that is not that of the master file. The transactions may be in another sequence, or may be unsequenced. In contrast to the selective sequential method, the access mechanism will move *not* in an ordered progression but back and forth along the file. Here the index is used when transactions are processed immediately - i.e., there is not time to assemble files and sort them into sequence. It is also used when updating two files simultaneously. For example, a transaction file of orders might be used to update a stock file *and a* customer file during the same run. If the order was sorted to customer sequence, the customer file would be updated on *a selective sequential* basis and the stock file on a random basis. (Examples will be given in later segments.)

Note. In *c.i* and *c.ii* the ordered progression of the heads relies upon an orderly organisation of the data and no other program performing reads from the disk at the same time, which would cause head movement to other parts of the disk. In multi-user systems these things cannot always be relied upon.

d. Random files. Generally speaking the method of access to random files is RANDOM. The transaction record keys will be put through the *same* mathematical formula as were the keys of the master records, thus creating the appropriate bucket address. The transactions in random order are then processed against the master file, the bucket address providing the address of the record required.

Methods of addressing

For direct access one must be able to "address" (locate) each record whenever one wants to process it. The main methods of obtaining the appropriate address are as follows:

a. *Index*: The record keys are listed with the appropriate disk address.The incoming transaction record key is used to locate the disk address of the master record in the index. This address is then used to locate the appropriate master record.

b. Address generation: The record keys are applied to a mathematical formula that has been designed to generate a disk hardware address. The formula is very difficult to design and you need not worry about it. The master records are placed on the disk at the addresses generated. Access is afterwards obtained by generating the disk address for each transaction.

c. *Record key = disk address*: It would be convenient if we could use the actual disk hardware address as our record key. Our transaction record keys would then also be the appropriate disk addresses and thus no preliminary action such as searching an index or address generation would be required in order to access the appropriate master records. This is not a very practical method, however, and has very limited application.

Updating Sequential files

a. The method of updating a sequential file is to form *a new* master each time the updating process is carried out. (The method applies to magnetic tape where the new file is written onto a different reel.)
b. Updating a master file entails the following:
    i. Transaction file and master file must be in the same sequence.
    ii. A transaction record is read into main storage.
    iii. A master record is read into main storage and written straight out again on a new file if it does not match the transaction. Successive records from the master file are read (and written) until the record matching the transaction is located.
    vi. The master record is then updated in storage and written out in sequence $\mathbf{o_n}$ the new file.The four steps are repeated until all the master records for which there is a transaction record have been updated. The result is the creation of a *new* file containing the records that did not change plus the records that have been updated. The new reel will be used on the next updating run.

Sequential file maintenance

File maintenance is the term used to describe the following:

a. Removing or adding records to the magnetic file.
b. Amending static data contained in a record, e.g., customer name and address, prices of stock items following a general price change.

The term generally applies to master files.

Removing a record entails leaving it off the carried-forward file, while adding records entails writing the new record onto the C/F file in its correct sequence. Variable-length records present no problems because space is allocated as it is required.

File labels

In addition to its own particular "logical" records (i.e., the customer or payroll records) each *file* will generally have two records, which serve organisational requirements. They are written onto the file in magnetic form as are the logical records. These two records are usually referred to as labels. One comes at the beginning of the file and the other at the end. This applies to magnetic tape too.

Header label. This is the first and its main function is to identify the file. It will contain the following data:

i. A specified field to identify the particular record as

a label.

ii. File name - e.g., PAYROLL; LEDGER; STOCK.

iii. Date written.

iv. Purge date - being the date from which the information on the particular file is no longer required and from which it can be deleted and the storage space re-used. This label will be checked by the program before the file is processed to ensure that the correct tape has been opened.

b. Trailer label. This will come at the end of the file and will contain the following data:

i. A specific field to identify the particular record as a label.

ii. A count of the number of records on file. This will be checked against the total accumulated by the program during processing.

iii. Volume number if the file takes up more than one cartridge or pack (or tape).

Control totals

Mention is made here of one further type of record sometimes found on sequential files - one which will contain control totals, e.g., financial totals. Such a record will precede the trailer label.

Buffers and buffering

The area of main storage used to hold the individual blocks, when they are read in or written out, is called a buffer. Records are transferred between the disk (or tape) unit and main memory one complete block at a time. So, for example, if the blocking factor is 6, the buffer will be at least as long as 6 logical records. A program that was processing each record in a file in turn would only have to wait for records to be read in after processing the sixth record in each block when a whole block would be read in. The use of just one buffer for the file is called single buffering.

In some systems double buffering is used. Two buffers are used. For the sake of argument call them A and B and assume that data is to be read into main storage from the file. (The principle applies equally well to output.) When processing begins the first block in the file is read into buffer A and then the logical records in A are processed in turn. While the records in A are being processed the next block (block 2) is read into B. Once the records in A have been processed those in B can be processed immediately, *without waiting for a read.* As these records in B are processed the next block (block 3) is read into A replacing what was there before. This sequence of alternately filing and processing blocks carries on until the whole file has been processed. There can be considerable saving in time through using double buffering because of the absence of waits for block reads.

Note: Single and double buffering are generally carried out by the operating system not by the application program.

Week Twelve

Non-sequential updating of disk files, File reorganisation, physical file organizations, File access methods and File calculations

Objective:

- To introduce students to file creation and maintenance in different data processing environment as the fundamental building blocks of knowing the relevant of computer file.
- To explore the file organisation design concepts including overflow handling in file storage and pitfalls.

Description: Serial Sequential file organisation, Index Sequential Organisation, Random file organisation and their access method were illustrated and discuss with an emphasis on the storage media.

Non-sequential updating of disk files

As individual records on disks are addressable, it is possible to *write back* an updated record to the same place from which it was read. The effect is therefore to *overwrite the* original master record with the new or updated master record. This method of updating is called "Updating in place" or "overlay". Note the sequence of steps involved:

a. The transaction record is read into main storage.
b. The appropriate master record is located on disk and is read into main storage.
c. The master record is updated in main storage.
d. The master record (now in updated form) is written from main storage to its original location, overwriting the record in its pre-dated form.

a. The method described can only be used when the *address* of the record is *known,* i.e., when a file is organised as indexed sequentially or randomly.
b. Files organised on a serial or sequential basis are processed in the same way as magnetic tape files, i.e., a physically different carry-forward master file will be created each time the file is processed. Such a new file could be written onto a different disk pack or perhaps onto a different area of the same disk pack.

File reorganisation

As a result of the foregoing the number of records in the overflow area will increase. As a consequence the time taken to locate such a record will involve first seeking the home track and then the overflow track.

Periodically it will be necessary to reorganise the file. This will entail rewriting the file onto another disk:

i. Putting the records that are in the overflow area in the home area in the proper sequence.
ii. Leaving off the records that have a deletion marker on them.
iii. Rewriting any index that is associated with the file.

Further details of physical file organisation

Sequential file organisation: The physical order of sectors, tracks and cylinders in which blocks are written, (and therefore subsequently read) is defined so as to minimise access times. This means that all sectors within the same cylinder are written to before moving to the next cylinder so as to minimise head movement. (Working from surface to surface on the same cylinder does not require movement of the read-write heads.) It also means that the sectors within a track are written in an order that reduces rotational delay. Ideally, this would

mean that the sectors are written (and read) in numbered sequence 1, 2, 3 etc but normally delays in the software or hardware controlling the reads and writes mean that one or more sectors have to be skipped between writes. For example, if there are 8 sectors on a track, the order of reads might be 1, 4, 7, 2, 5, 8, 3, 6 with a delay of two sectors between each read.

Index Sequential Organisation: The same principles apply with respect to minimising access time but the situation is complicated by the more complex organisation of the file.

The index sequential file is created from a file that is already in sequential order. The indexes are generated and included as the index sequential file is organised and stored. The indexes are subsequently updated as the file is updated.

a. The primary index is created in main storage as the file is organised, and stored on the disk when organisation and storage of the file is completed. It is loaded into main storage again at the start of any subsequent access. The primary index is normally organised as a sequential file on its own area of the disk e.g. on its own cylinder.
b. A secondary index is also created in main storage while each cylinder is organised and stored. There is one secondary index per cylinder. During the organisation of each cylinder provision is made for local overflow, i.e., the provision of spare storage on the *same* cylinder, for tracks that subsequently may become full and therefore unable to accommodate further records. (Global overflow, i.e., overflow of the whole file, may be catered for in a similar way.)
c. Space is left on each surface during the initial organisation so that a few additions can be made before overflow occurs.
d. When the file is accessed, in the absence of any overflow, the access times for primary index, secondary index and the data access itself are kept to a minimum. Performance degrades as the amount of overflow increases because of the additional index references incurred.

Random file organisation: In this method the keys are used to allocate record positions on the disc. For example, a record whose key was 149 could be allocated the position surface 1 track 49. We say that the disk address of the record has been *generated* from the key and so the technique is called address generation. The generated disk address usually gives just enough detail to specify the block in which the record is to be placed or found; so that when a record is to be accessed the whole block is input, and the record is searched for within the block. We thus have organisation by address generation and access by address generation. Sometimes an index of generated addresses is produced as the file is created. This index is then stored with the file. It is then possible to access the file by means of this random index. We then have organisation by address generation and access by random index.

Hashed keys: When disk addresses are generated directly from keys, as in the example just given, there tends to be an uneven distribution of records over available tracks. This can be avoided by applying some algorithm to the key first. In this case we say the key is hashed. Examples

a. Squaring, e.g., for key number 188

188^2 =	35	3	4	4
DISC ADDRESS	Track Number	Surface Number	Bucket Number	Block Number

Fig 10: Disc organisation Structure

b. Division method, e.g., for key number 188.
188 ÷ 7 = 26 Remainder 6. So we could use track 26 surface 6 say.
Hashing reduces the chances of overflow occurring, but when overflow does occur records are normally placed on the next available surface in the same cylinder so as to minimise head movement.

Other organisation and access methods

The methods of file organisation and access described so far are closely related to the physical features of the disk. It is now common practice for much of this detail to be "hidden" from programmers or users of the system. Instead the operating system handles the physical levels of organisation and access, and provides standard *logical file* organisation and access methods.

Some examples of logical file organisation.
a. *Sequential files*: A programmer need merely regard the file as a sequence of records, and need have no concern for their physical location. A program instruction to read the next record will result in the appropriate record being transferred into memory, i.e., the programmer's "view" may just be like this.

R1	R2	R3	R4	R5	R6	etc

R1…. R2 are logical records.

b. *Direct files*. These are files that provide fast and efficient direct access, i.e., they are
normally *random files* with *one* of a number of appropriate addressing methods. A common type of direct file is the Relative file. The logical organisation of a relative file is like this:

R1	R2	R3	R4	R5	R6	etc
1	2	3	4	5	6	etc

RI.... R6 are *logical records* with *logical* keys 1....6.
A relative file may be accessed sequentially or randomly.

c. *Index sequential files*. Logical versions of index sequential files are simpler than their physical counterparts, in that logical keys are used instead of disk addresses, and details of overflow are hidden from the programmer.

File calculations

Two basic types of calculation are often needed when using files:

a. The storage space occupied by the file (for magnetic tape the length of tape may be required).
b. The time taken to read or write the file.

A simple example now follows.

For a sequential file on disk the basic calculation for estimating the required space is as follows.

a. Divide the block size by the record size to find how many *whole records* can fit into a block. This is the blocking factor.
b. Divide the total number of records by the blocking factor to obtain the total number of blocks required.
c. Multiply the block size in bytes by total number of blocks required.
Note. This basic method can be modified if the records are variable in length (e.g. use an average).

For non-sequential files on disk the storage space required is greater than that for sequential because of the space allowed for insertions and overflow. The exact calculations depend on the software used to organise the files and on the ways in which it is possible to configure the settings. However, a typical overhead is 20% more than that for sequential.

We may deduce that

average access time = seek time + latency + data transfer time.

For a random file where N records were read the total read time would simply be

N x average access time.

Note that a whole sector would have to be read in just to get each individual record.

For a sequential file, where the disk was not being used for any other accesses at the same time there would be a seek for each cylinder and then all sectors in the cylinder would be read. This suggests a formula such as:

Total read latency + data number of seek time + x sectors per x cylinders time =transfer time cylinder per file

A review of storage methods

Having examined file organisation and access we are now in a better position to review files storage methods. The main types of storage are:

a. (IAS) **Immediate-access** storage e.g., RAM
b. (DAS) Direct-access storage e.g., disk
c. (SAS) Serial-access storage e.g., magnetic tape

The tasks that they are best suited to carry out are:

a. *Immediate access***:** Because of its unique electronic properties giving extremely quick access to stored data this type of storage is used as the computer's main storage. Access to individual characters/bytes is completely independent of their position in store. It is in main storage that the programs are held during processing so that their instructions can be executed swiftly. Also the particular data currently being worked on is held in main storage. Ideally, IAS would be used for storage of all data because it is fast in operation. This is not practicable because of cost and volatility, and its use therefore is limited to main storage. Thus some alternative form must be found for storing the files and data which are not immediately required by the program.

b. *Direct access:* One such alternative is DAS (e.g. disk storage). Storage capacity can vary from thousands to hundreds of millions of characters. DAS has the important facility of allowing direct access, that is, records can be accessed independently of each other. It is thus suitable for files being processed in a selective manner and also for storing programs, and systems software that are required to be called into main storage at any time during the running of an application program. It is an essential requirement for on-line processing or where file-interrogation facilities are needed.

c. *Serial access:* If bulky auxiliary storage is required and the need for random access is not present then the choice may well fall on SAS, the most common form of which is magnetic tape. It is also cheaper than main memory or disk. Because of its inherent serial nature, access to a record on tape is a function of its position on the tape. Therefore every preceding record on a master file must be read (and written out onto the new tape) before the required record can be located. Nevertheless millions of characters can be stored very cheaply.

Summary

a. File maintenance involves adding and deleting records and the amendment of static data contained in records.
b. Labels are provided for control and organisational purposes.
c. Conceptually the disk is regarded as being composed of so many concentric CYLINDERS.
d. Disk is an addressable medium and therefore specific records can be accessed leaving the rest of the file undisturbed.
e. Organisation on disk is by cylinder, track and bucket (or block).
f. Access time on disk consists of three components seek time, rotational delay and data transfer time.
g. The overlay or "in-place" method of updating *can* be used on disk.
h. Methods of file organisation on disk are:
    i. Serial.
    ii. Sequential (with or without an index).
    iii. Random.
i. Methods of access to disk files are:
    i. Serial (for serial and sequential files).
    ii. Selective sequential (for indexed sequential files).
    iii. Sequential (for sequential and indexed sequential files).
    iv. Random (for random and indexed sequential files).
j. File maintenance on disk raises problems:
    i. When inserting new records.
    ii. When updating variable-length records.
k. Special overflow areas are designated on disk to take new records and overlength records temporarily.
1. A periodic file re-organisation run is required with disk files to place records residing temporarily in overflow areas into their correct sequence in the file and to leave off "deleted" records. Any index also has to be reconstructed during this run.
m. A summary of normally accepted methods of file organisation on

disk and associated
methods of access is given:

FILE ORGANISATION METHOD	METHOD OF ACCESS
l. Serial (Sequential)	Serial (Sequential)
2. Sequential	Serial (Sequential)
3. Indexed sequential	a. Sequential b. Selective sequential c. Random (Direct)
4. Random (direct or relative)	Random (Direct)

Fig 10: A summary of file organisation and access methods.

Week Thirteen and Fourteen

Revisions and Examinations Objective:
The objective of the week lecture is for the student to be able to revise all they have been taught so far. **Description:** All the objectives for the course should be seriously overviewed