UNIT 4

FILE HANDLING IN COBOL

FILES

A record is a group of logically or functionally related fields.

A File is a group of Records.

A group of records, which can be created, copied, modified, retrieved and deleted.

E.g.: Details of an employee
-Name, Adds, Phone no., Dept no etc… Forms a record

Details of all employees
-Group of such record forms a file.

Figure 4-1 FILES

Notes:

Files can be broadly categorized into Program files and Data files. In COBOL the term "Files" is used to indicate data files. Data files are normally created on a tape or disk and subsequently program can refer them.

Fixed vs Variables Length Records

Fixed length records.

- Corresponding fields of all the records have same length.

Variable length records.

- Field lengths may vary from record to record.

Figure 4-2 Fixed vs Variable Length Records

Notes:

The size of a record is the cumulative size of all the fields in it.
If all the records of a file have the same structure then they are called Fixed length-records. For convenience, records of different lengths can be placed together in one file. Then they are known as variable-length-records.

FILE-CONTROL Paragraph

Format:

```
SELECT [OPTIONAL] File-name-1  ASSIGN  TO  Assignment-name-1
[ RESERVE                      <INTEGER> AREA ]
                             { SEQUENTIAL  }
[ ORGANIZATION IS            { INDEXED     }  ]
                             { RELATIVE    }

                             { SEQUENTIAL }
[ACCESS MODE  IS             { RANDOM     }  ]
                             { DYNAMIC    }

[FILE STATUS IS Data-name-1]
```

Figure 4-3 FILE-CONTROL - SEQUENTIAL

Notes:

The FILE-CONTROL paragraph associates each file with an external data-set. FILE_CONTROL paragraph is in INPUT-OUTPUT Section of ENVIRONMENT Division.Not all options are available on all platforms.

SELECT OPTIONAL may be specified only for files opened in the input, I-O, or extended mode. You must specify SELECT OPTIONAL for such input files that are not necessarily present each time the program is executed.

The ‘file-name-1’ must be identified by an FD or SD entry in the DATA DIVISION.

The ASSIGN clause associates the program’s name for a file with the external name for the actual data file.

The RESERVE clause allows you to specify the number of input/output buffers to be allocated at run time for the file.

The ORGANIZATION clause identifies the logical structure of the file.

ORGANIZATION IS SEQUENTIAL

The Records are stored in contigious allocation. To access the record in Sequential mode only(I,e to read the last record, it reads all the records until last record found.) Deletion of record is not possible.Updation is possible but record length should not changed.

ORGANIZATION IS INDEXED

Each record in the file has one or more embedded keys; each key is associated with an index. An index provides a logical path to the data records, according to the contents of the associated embedded record key data items. Indexed files must be direct-access storage files. Records can be fixed-length or variable-length.
Each record in an indexed file must have an embedded prime key data item. When

records are inserted, updated, or deleted, they are identified solely by the values of their prime keys. Thus, the value in each prime key data item must be unique and must not be changed when the record is updated.

In addition, each record in an indexed file can contain one or more embedded alternated key data items. Each alternated key provides another means of identifying which record to retrieve.
The RECORD KEY clause specifies the data item within the record that is the prime RECORD KEY for an indexed file. The values contained in the prime RECORD KEY data item must be unique among records in the file.
The ALTERNATRE RECORD KEY clause specifies a data item within the record that provides an alternated path to the data in an indexed file. Used like the RECORD KEY but for an alternate index.

ORGANIZATION IS RELATIVE

The INPUT-OUTPUT FILE-CONTROL for Relative record files is very similar to that of indexed files except you use the RELATIVE KEY clause of the ACCESS MODE phrase and each record identified by the Relative Record Number instead of Recoed Key.

ACCESS Mode

Modes	**Meaning**
SEQUENTIAL	Records of the file can be accessed sequentially, starting from first record till the required record is reached.
RANDOM	Any record can be accessed directly without beginning from the first record.
DYNAMIC	Records can be accessed both randomly and/or sequentially

Figure 4-4 ACCESS modes

Notes:

The record of a file stored on a magnetic tape can be accessed in sequential mode only. But the records of file stored on magnetic disk can be accessed in all the modes.

FILE STATUS Clause

A two-digit number indicates the status of the file.

Value	Status
00	Successful Completion
10	At end condition
30	Permanent error
34	Boundary violation

Figure 4-6 FILE STATUS

Notes:

Input-Output operations may not be successful thus resulting in termination of the program.

The data-name specified in the file-status clause contains the status code and can be referred by the programmer. Depending on the code programmer can take specific actions by transferring the control to error-routine paragraphs.

The data name should be declared in working-storage section with alphanumeric data type of two characters.

I-O-CONTROL Paragraph

- **The Optional I-O-CONTROL paragraph of the Input-Output Section specifies when checkpoints are to be taken and the storage areas to be shared by different files.**

- **Specifies information needed for efficient transmission of data between the external data set and the COBOL program.**

Figure 4-7 I-O-CONTROL Paragraph

Notes:

The I-O-CONTROL paragraph is optional.

The key word I-O-CONTROL can appear only once, at the beginning of the paragraph. The word I-O-CONTROL must begin in Area A, and must be followed by a separator period.

Each clause within the paragraph can be separated from the next by a separator comma or a separator semicolon. The order in which I-O-CONTROL paragraph clauses are written is not significant. The I-O-CONTROL paragraph ends with a separator method.

FILE SECTION

```
FILE SECTION.

FD File-Name
        BLOCK CONTAINS  m RECORDS
        RECORD CONTAINS n CHRACTERS
        LABEL RECORDS ARE STANDARD/ OMMITED
01      File-record-structure.
```

Figure 4-8 FILE STATUS

Notes:
Each file used in the program should have an FD entry (File Description) in FILE SECTION.

- BLOCK CONTAINS clause specifies number of records in the block.
- RECORD CONTAINS clause specifies total number of characters in each record.
- LABEL RECORDS clause indicates
    - Disk files if STANDARD option is specified
    - Print files if OMITTED option is specified
- Value clause specifies the name of the physical file and the path
- 01 level entry should follow immediately after FD paragraph.

Blocking
Input-Output operations are slower compared to CPU processing speed. To reduce the CPU waiting time, block of records from the disk can be moved to the memory space called buffer thus reducing number of I-O operations.
The Programmer can specify the number of records contained in a block. Suitable block size is to be selected by the programmer.

File Operations

Cobol Verbs	**Meaning**
WRITE	Writes the records into file. Required while creating a new file and while Adding new records to an existing file.

Figure 4-9 File Operations

Notes:

This foil lists the possible operations that can be performed over files.

Before doing any operation, files should be opened and they must be closed before exiting the program
"OPEN" and "CLOSE" verbs are provided by COBOL.

OPEN modes

Mode	**Meaning**
INPUT	Stands for input mode. Only reading of records possible.
OUTPUT	Stands for output mode. Only writing new records possible.
I-O	Stands for Input ---Output mode.

Figure 4-10 OPEN modes

Notes:

SYNTAX

OPEN Mode File-name1, File-name2.
CLOSE File-name1, File-name2

While opening the file the mode must be specified depending on the operation to perform.

More than one file can be opened and closed. Further, files can be opened and closed more than once in a program.

READ – Sequential Access

- **When the READ statement is executed the file must already be open in INPUT or I-O mode**
- **The AT END clause must be before the NOT AT END**

Format 1: sequential retrieval

```
>>____READ__file-name-1___   __________________  __  ___________  ____________________>
                             |_ NEXT ___________|    |_RECORD__|
                             |_              (1)|
                             |_  PREVIOUS______|

>_____  ______________________    _______________________________________________>
       |____ INTO___identifier-1____|

>_____  ________________________________    ________________________________>
       |_    ____    __END_imperative –statement-1_|
          |_ AT _|
>_____  ______________________________________    ____    __________    _______><
       |_ NOT___   ______   ___END_imperative-statement-2_|        |_END-READ_|
```

Figure 4-12 FILE STATUS

Notes:
For sequential access, the READ statement makes the next logical record from a file available to the object program. For random access, the READ statement makes a specified record from a direct-access file available to the object program.

When the READ statement is executed, the associated file must be open in INPUT or I-O mode.

NEXT RECORD Reads the next record in the logical sequence of records. NEXT is optional when ACCESS MODE IS SEQUENTIAL;

PREVIOUS RECORD Reads the previous record in the logical sequence of records.

END OF FILE Processing

- **When the 'AT END' condition occurs during sequential processing, the READ statement execution is unsuccessful. The contents of the record area are 'undefined'**

- **The following actions take place when 'AT END' occurs:**

- **The status indicator is posted.**
- **Control is transferred to the AT END phrase, if it is specified**
- **If AT END is not specified, then USE AFTER STANDARD ERROR could be specified and that procedure is executed. Then control is returned to the statement following the READ.**

Figure 4-13 END OF FILE Processing

Notes:

READ – Random Access

Format 2 : Random Retrieval

```
>>_____READ______file-name-1____   __________  ___  _______________   _____________>

>____   _____________________________    ________________________________________>
        |_KEY_____    ___   __data-name-1__|
                   |_TO_|

>_____   _______________________________________________  _______________________>
         |_INVALID_____    ______   ____imperative-statement-3__|
                       |_KEY __|

>____   ____________________________________________  ___   ____________  _______><
        |_NOT INVALID___   ___   __imperative-statement-4_|     |_END-READ__|
                         |_KEY_|
```

- **For VSAM INDEXED files, the KEY field contains a data value that will be matched against the key filed in the file records until the first record having an equal value is found.**
- **For VSAM RELATIVE files, the KEY phrase must not be specified.**

Figure 4-14 END OF FILE Processing

Notes:

Format 2 must be specified for indexed and relative files in random access mode, and also for files in the dynamic access mode when record retrieval is random.

Execution of the READ statement depends on the file organization.

Indexed Files

Execution of a Format 2 READ statement causes the value of the key of reference to be compared with the value of the corresponding key data item in the file records, until the first record having an equal value is found. The file position indicator is positioned to this record, which is then made available. If no record can be so identified, an INVALID KEY condition exists, and READ statement execution is unsuccessful.

If the KEY phrase is not specified, the prime RECORD KEY becomes the key of reference for this request. When dynamic access is specified, the prime RECORD KEY is also used as the key of reference for subsequent executions of sequential READ statements, until a different key of reference is established.

Relative Files

Execution of a Format 2 READ statement sets the file position indicator pointer to the record whose relative record number is contained in the RELATIVE KEY data item, and makes that record available.

The KEY phrase must not be specified for relative files.

READ – Dynamic Access

- **For dynamic access, either sequential or random access possible, depending upon the format of the Read statement**
- **Dynamic access is allowed only for VSAM indexed or VSAM relative organizations.**

- **Dynamic access is established by ACCESS IS DYNAMIC in FILE-CONTROL SELECT statement**

- **The NEXT phrase must be specified for sequential access with dynamic mode. In order to READ NEXT, "position" must have been established in the file by a successful OPEN, START or READ statement**

Figure 4-15 READ – Dynamic Access

Notes:

START Statement

Format :

```
>>___START___file-name-1___________________________________________________________>

>__  ___________________________________________________________________   _________>
    |_KEY___  ______  ____  __EQUAL___  ___  _________________ _data-name-1____|
             |__TO  _|      |           |_ TO_|                |
                            |_ = _______________________________|
                            |_LESS__  _______  _________________|
                            |         |_THAN_|                  |
                            |_ <  ______________________________|
                            |_GREATER__  ____     ______________|
                            |            |_THAN_|               |
                            |_> ________________________________|
                            |_NOT LESS___  _______   ____________|
                            |              |_THAN _|             |
                            |_NOT < _____________________________|
                            |_NOT GREATER__  _______   ________|
                            |                |_THAN_|          |
                            |_NOT > ___________________________|
                            |_LESS_  ____ _ OR EQUAL_  __  _____|
                            |       |THAN|             |_TO_|   |
                            |_ < =  ____________________________|
                            |_GREATER__  ____  _OR EQUAL_ __ _|
                            |            |_THAN_|           \ TO|
                            |_>+________________________________|

>__  _______________________________________   ____________________________________>
    |_INVALID___  _____ _imperative-statement-1_|
                 |_KEY_|

>__  _______________________________________   ___________  ____________  ______________>
    |_NOT INVALID___  _____  imperative-statement-1_|        |_END-START_|
                    |_KEY_|
```

Figure 4-16 START Statement

Notes:
The START statement provides a means of positioning within an indexed or relative file for subsequent sequential record retrieval.
When the START statement is executed, the associated indexed or relative file must be open in either INPUT or I-O mode.

file-name-1
Must name a file with sequential or dynamic access. File-name-1 must be defined in an FD entry in the Data Division, and must not name a sort file.

END-START Phrase

This explicit scope terminator delimits the scope of the START statement. END-START converts a conditional START statement to an imperative statement so that it can be nested in another conditional statement. END-START can also be used with an imperative START statement.

WRITE Statement

- **The WRITE statement releases a logical record for an output or input/output file.**

- **When the WRITE statement is executed:**
  - **The associated sequential file must be open in OUTPUT or EXTEND mode.**
  - **The associated indexed or relative file must be open in OUTPUT, I-O, or EXTEND mode.**
- **Record-name must be defined in a Data Division FD entry. Record-name can be qualified. It must not be associated with a sort or merge file.**

Figure 4-17 WRITE Statement

Notes:

WRITE……….FROM

PROCEDURE DIVISION.

WRITE File-rec FROM Identifier.

- **File-rec is record-name declared in FILE-SECTION.**
- **Identifier is a working-storage section variable**
- **The length of the identifier should be equal to the length of the record.**

Figure 4-18 WRITE …FROM

Notes:

To Create a file, program can accept the data from the terminal into file record and write it.

If the data need to be processed, it can be accepted in a W-S identifier. After processing the data the above WRITE……..FROM statement can be issued.
Each WRITE statement writes one record at a time.

READ…………INTO

PROCEDURE DIVISION.

READ FILE-name (INTO W-S-Rec) | (AT END Statement)

- File name is defined in SELECT clause.

Figure 4-19 READ……INTO

Notes:
READ statement on sequential files reads one record at a time and makes it available to program.

Reading begins from first record and if the READ statement is put in a loop
That is executing the statement repeatedly, then it is possible to read consecutive records.

Loop can be terminated before AT END condition is reached if required so by the program.

If the file is left open next time when the read statement executes, reading continuous from where it was stopped before the termination of loop.
If the file is closed then it is to be opened again before reading it.

REWRITE & DELETE

REWRITE record-name (FROM identifier)

Updates an existing record from W-S identifier.

OPEN I-O File-name

DELETE record-name ----------------→not allowed

Figure 4-20 REWRITE & DELETE

Notes:
It is often required to change the existing data and the process is called **UPDATING.** COBOL provides REWRITES verb to modify an existing record.

For example, changing the address field of an employee requires reading of employee number. Every record to be updated needs to be read first. To search the record of an employee, whose employee number is known, the process is as follows

- Store the employee number in a variable
- Open the file
- Read first record
- Compare the variable with Emp-No field of the file
- If it matches update his address by REWRITE
- Else read next record his address by REWRITE
- Repeat the process until the require record is read.

Appending to sequential files

Adding new records to the existing file.

OPEN EXTEND Mode

WRITE records.

When new records to be added to file open the file in EXTEND mode EXTEND mode causes the pointer to move to the end of the file.

Figure 4-21 Appending to sequential files

Notes:

CLOSE Statement

Format :

CLOSE File-name-1, [File-name-2 ………….]

CLOSE Statement Releases the Resourcces which are assigned to that file.

Cannot Close the file which is not opened.

After performing the operations on the file (I,e no longer used in a program) needs to be closed but not necessary.

If the FILE STATUS clause is specified in the FILE-CONTROL entry, the associated status key is updated when the CLOSE statement is executed.

If the file is in an open status and the execution of a CLOSE statement is unsuccessful, the EXCEPTION/ERROR procedure (if specified) for this file is executed.

Figure 4-23 CLOSE

Notes:

Sequential Files

```
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
          SELECT file-name ASSIGN TO  DEVICE-NAME
          ORGANIZATION IS SEQUENTIAL.
          ACCESS MODE IS SEQUENTIAL.                    Area
          FILE STATUS IS data-name.                      B
```

Figure 4-24 Sequential Files

Notes:

All the files used in the program should have an entry in FILE CONTROL

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  SEQFILE.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT  SEQ-FILE ASSIGN TO ‘DD-NM1’
                ORGANIZATION IS SEQUENTIAL.
        SELECT  OUT-FILE ASSIGN TO ‘DD-NM2’
                ORGANIZATION IS SEQUENTIAL.

DATA DIVISION.
FILE SECTION.
FD  SEQ-FILE.
01 SEQ-REC.
             02      NO        PIC  X(3).
             02     NAME    PIC  X(15).
              02    ADDR      PIC X(10).
FD  OUT-FILE.
01 OUT-REC                PIC  X(28).
WORKING-STORAGE SECTION.
01      EOF         PIC  X.
PROCEDURE DIVISION.
MAIN-PARA.
          OPEN INPUT SEQ-FILE
                     OUTPUT OUT-FILE.
          READ  SEQ-FILE INTO  OUT-REC AT END MOVE ‘Y’ TO EOF.
          PERFORM READ-PARA UNTIL EOF=’Y’.
          CLOSE SEQ-FILE
                        OUT-FILE.
           STOP RUN.
READ-PARA.
             WRITE OUT-REC.
             READ SEQ-FILE INTO OUT-REC AT END MOVE ‘Y’ TO EOF.
```

EXA

Indexed Files

Index component consists of a index structure with a record key values and addresses of corresponding records.

RECORD KEY is one or more fields of the records.
Suitable record key is to be chosen by the programmer depending on the functionality of the fields.

E.g : Employee-code, Job-number.

ALTERNATE RECORD KEY can also be chosen.

E.g : Employee-name, Job-name.

Indexed files facilitate faster accessing of records compared to that of sequential files.

Figure 4-25 Indexed Files

Notes:
When an indexed file is created

- An index component is also created containing some index tables based on record keys.

- A data component is created containing the actual records.

Record keys identify every record in the file.

The process of accessing a record involves searching for the record key with matching index value. Then locate the record from the corresponding address.
This is done by the system itself.

INVALID KEY

(READ | WRITE | REWRITE | DELETE) File-name
(INVALID KEY Statement)
(AT END Statement).

Records and indexes of an indexed file are stored in key sequence order to facilitate faster access.

Invalid key clause checks whether any input-output operation is violating the

Uniqueness of primary keys
E.g. add a record with duplicate value.

Sequence of the records.
E.g. add a record with key value out of range.

Proper read
E.g. try to read a non-exist record

Reading of a record in indexed files required the key value to be provided by the program

Figure 4-26 INVALID KEY

Notes:

ACCESS MODE: SEQUENTIAL & RANDOM

ACCESS MODE SEQUENTIAL

- READ File-name NEXT RECORD to read sequentially.
- DELETE statement should not contain invalid key

- AT END clause is required.

ACCESS MODE RANDOM

- READ File-name INVALID KEY statement
- AT END clause not required.

Figure 4-27   ACCESS MODE : SEQUENTIAL & RANDOM

Notes:
When READ NEXT statement is to be executed each time the records are read consecutively.

IF the access mode is RANDOM a record is read from corresponding key value.

ACCESS MODE: DYNAMIC

ACCESS MODE DYNAMIC

START file-name key
(NOT | LESS THAN | GREATER THAN| LESS THAN) identifier
INVALID KEY statement
READ file-name NEXT RECORD AT END statement.

Figure 4-28 ACCESS MODE : DYNAMIC

Notes:

In a situation demanding the access of more than one consecutive records from the middle of the file then dynamic access is used.

The “START” verb places the read pointer to the record whose key value is compared with an identifier. Record is accessed randomly.

‘READ…NEXT’ can be put into loop for sequential reading.

For the Rewrite/ Delete operations the records must be read at first.

EXAMPLE: INDEXED FILE

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  SEQFILE.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
        SELECT  IND-FILE ASSIGN TO ‘DD-NM1’
                ORGANIZATION IS INDEXED
                ACCESS  MODE IS DYNAMIC
                RECORD KEY IS NO.
        SELECT  OUT-FILE ASSIGN TO ‘DD-NM2’
                ORGANIZATION IS INDEXED
                 ACCESS MODE IS DYNAMIC.
                 RECORD KEY IS NO.
DATA DIVISION.
FILE SECTION.
FD  IND-FILE.
01 IND-REC.
             02     NO          PIC  X(3).
             02     NAME     PIC  X(15).
             02     ADDR      PIC X(10).
FD  OUT-FILE.
01 OUT-REC.
            02     NO        PIC  X(3).
            02     NAME    PIC  X(15).
            02    ADDR      PIC X(10).

WORKING-STORAGE SECTION.
01      VAL        PIC  X(3).
PROCEDURE DIVISION.
MAIN-PARA.
          OPEN INPUT IND-FILE
                    OUTPUT OUT-FILE.
          MOVE ‘001’ TO VAL.
          READ  IND-FILE RECORD KEY VAL INTO  OUT-REC INVALID KEY GO TO
                                                                    ERR-PARA.
         WRITE OUT-REC.
          PERFORM EXIT-PARA.
ERR-PARA.
         DISPLAY ‘KEY NOT FOUND’.
         PERFORM EXIT-PARA.
EXIT-PARA.
        CLOSE IND-FILE  OUT-FILE.
        STOP RUN.
```

Relative Files

```
FILE CONTROL
        SELECT file-name ASSIGN TO Disk
        ORGANIZATION IS RELATIVE
        RELATIVE KEY  data-name-1
```

RRN indicates the offset of a record from the first record of the file.

Figure 4-29   ACCESS MODE : DYNAMIC

Notes:

In relative file **Relative Record Number** identifies the records of the file. Select clause should specify “RELATIVE KEY”.

Value of data-name-1 indicates RRN.

Usage of READ/WRITE/ REWRITE/ DELETE statements, ACCESS modes, OPEN modes and START verb, are exactly similar to that for sequential files.

		Open modes									
		File Organization									
		Sequential				Relative			Indexed		
ACCESS MODE		OPERATIONS	I	O	I-O	E	I	O	I-O	I	O	I-O
	SEQUENTIAL	READ	X		X		X		X	X		X
		WRITE		X		X		X			X	
		REWRITE			X				X			X
		START					X		X	X		X
		DELETE							X			X
	RANDOM	READ					X		X	X		X
		WRITE						X	X		X	X
		REWRITE							X			X
		START										
		DELETE							X			X
	DYNAMIC	READ					X		X	X		X
		WRITE						X	X		X	X
		REWRITE							X			X
		START					X		X	X		X
		DELETE							X			X