Table of Contents:

File Processing

Hierarchy of Data

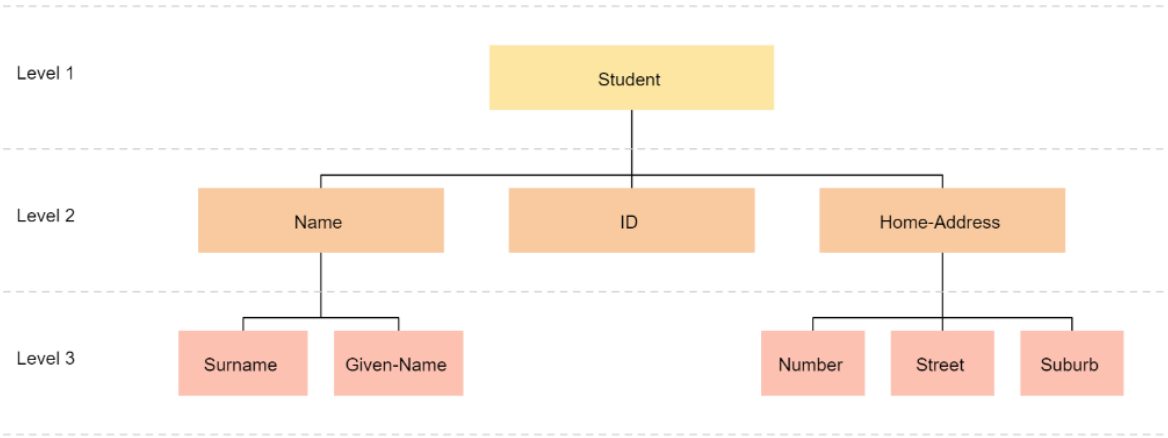
Introduction

Input used to be from punched card or punched paper tape files. Permanent files were maintained on magnetic tape and later magnetic disk. Output of information was directed to printed files.

This file orientation has been carried through the various revisions of COBOL and has been enhanced in later versions by syntax, enabling the manipulation of various models of databases, including the most common database type, relational.

Data Structure

Inherent in the syntax of COBOL is the means of processing data that is arranged in a hierarchical structure.



This means a reference to a higher level in the data structure automatically includes the data from the lower levels. If a reference is made to a Level 1 data item then any sub levels defined will be used in the COBOL statement.

Cobol's Basic Elements

A file is a collection of related records

- **Record:**
  - It is a self-contained set of data to one entity in a file
  - Each record consists of several fields
- **Field:**
  - A field is one item of data in a record e.g. a name
  - Each field consists of a number of characters
- **Character:**
  - A character is the lowest division of data normally addressable by a COBOL program
  - It is a letter, digit, or symbol able to be represented by the internal code used by the computer
  - This code will either be ASCII or EBCDIC
- **Binary Digits:**
  - Some earlier versions let the programmer address individual bits comprising of a character

Defining Data Structure

The records on a file are declared as Level 1 items. Their components are then given level numbers that reflect their position in the record and their subdivision into finer components where necessary.
This subdivision is determined by the designer of the data structure and it depends on the level of detail that the program needs to address to carry out its task.

File Concepts

Files can be classified by the functions that they perform in applications. The most commonly referred to files are:

- Master Files
- Transaction Files
- Reference Files

Master Files

It is a permanent collection of records related to one aspect of an organisation's activities or one set of commercial entities

Example:

- Organisation's customers
- Hospital's patients
- College's students

They are updated in an online, interactive mode or by a batch process. Transactions that take place during a period of time, perhaps a day, a week, or a month, are grouped and applied to the master file data at the end of the related period.

Example:

- Company might process purchases made online through the internet and process mail-order purchases in a daily batch process.

Transaction Files

These files contain records that are used to update master files

Example:

- An online customer places an order
- The customer orders file is a transaction file that contains all sales transactions for all customers
- The customer orders transaction file is used to update the customer master file by adjusting the customer's debit balance
- The customer orders transaction file also will be used to update the product master file by adjusting the figures for remaining stock on the shelf

Reference Files

This file provides information for a process without being part of the transaction being processed.

Example:

- A catalog file might provide the current price for a product being ordered from a supplier. In the process of updating the supplier master file with orders placed for products, each product ordered will require access to the catalog reference file to determine its current price.

In another process, the reference file will be regarded as the master file. In the previous example, the prices in the catalog file may need to be updated. This consists of a number of transactions. For example, price changes applied to the catalog, master, file.
Hence, a specific file can be treated as either a master or reference file depending upon its usage context.

Access to Files

Serial:

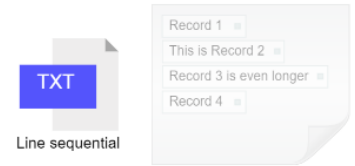
Serial access to a file involves starting at the first record on the file and progressing physically through the file in the order in which the records are encountered. This access method is often referred to as sequential access.

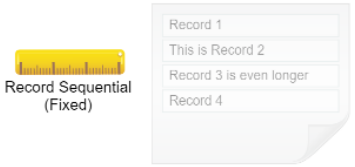
For a file stored on magnetic tape, serial access is the only way to access the records.

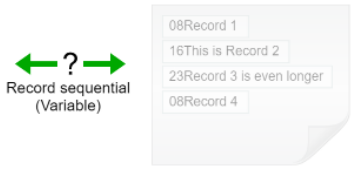
Sequential:

There are many uses for sequential files, including log files, tape files, printer output, and more. COBOL allows for two types of sequential file:

- Line sequential: each record in the file is terminated by a special character
- Record sequential: the file structure does not need a special character to terminate a record







Direct:

Direct access to records implies that a record can be located and accessed immediately, without searching through all the records individually. This is regardless of the position of the record in the file.

Direct access can only be performed when the file is stored on a device that allows it. Magnetic disks, optical disks, and solid state drives all allow direct access, magnetic tape does not.

Indexed Sequential:

Indexed files, also called Indexed Sequential, use a set part of each record as a key. For example, the first 10 bytes. The same part is used for every record.

In a separate area, each key and its location in the file is stored. This is called an index. A program can still access the file sequentially, or it can use a key to directly locate and access a single record.

A file could possibly have more than one index, in this example the Customer number, and the Customer name are indexes. A program could use either to locate a record. However at least one index must be unique, only one per file index. This is called the primary index.

Other indexes are called alternate indexes. There can be multiple records with the same alternate index. In these cases, program code must allow for the possibility of multiple matching records.

Relative/Random Files:

In Relative Files, records are identified by their position in the file. For example, the first record would be identified as 1, the tenth record as 10, and so on. Records can be processed sequentially, or a record can be accessed directly if the record number is known.

Relative records are normally of fixed length. Deleted records remain in the file, but are marked deleted. Relative files are sometimes called 'Random' files.

File Operations

Just like any other programming language, COBOL has operations to handle the processing of data from files. These operations include: OPEN, CLOSE, READ, and WRITE.

Syntax - Open:

When a program opens a file, it requests the operating system to locate a file, or create a new one. The program will often pass a directory name or location, and a file name. It may also pass to the operating system how the file will be used, for input, for output, or both.

The operating system will during this processing check for things like security access, and pass back return codes to the program indicating whether the Open was successful or not.

- A file cannot be accessed by a program until it has been opened.

In multiuser systems, the opening mode can also dictate whether the file is to be opened for exclusive use by the program opening it or whether the program is prepared to share it with other programs running at the same time.

Syntax - Close:

Closing a file releases its resources, file handle and buffers, from the program's control and no further action can be taken by the program with that file without the program reopening it.

Closing a file can also flush to the file any internal record buffers maintained by the program. If such buffers are maintained, failure to close a file can result in records that the program thinks have been written to the file failing to be physically output to that file. It is the programmer's responsibility to close any files their program has opened to eliminate this type of situation.

Syntax - Read:

A read operation will read one or more records from a file. A sequential read will read these records one at a time starting from the first record.

These records may be in a special sequence, or simply be in the sequence in which they were written to the file.

A direct Read addressed to a file locates a nominated record regardless of its physical position on the file. In such an operation, some means of locating the record must be given.

Syntax - Write:

Each Write operation places a nominated number of characters on a file. As for the Read operation, the Write typically consists of one record of the type stored on that file.

In writing records to a sequential file, each Write operation places a new record on the file. Do note that a Write operation for a file to which the program has direct access must differentiate between placing a new record on that file and rewriting an existing record after it has been read and updated.

An important process to note is the technique known as buffering or blocking records on a file.

Concept - Buffering:

Buffering consists of assembling a number of records in memory as the program "writes" them and actually places them on the file medium when a predetermined block size has been achieved.

This can make more efficient use of the physical storage medium than actually writing every record individually. The process is transparent to the program and requires no particular instructions on the part of the programmer.

However, when a program comes to the end of its operations and wishes to terminate, a partly-filled buffer might contain records that the program thinks it has written to the file but are still in internal memory.

Sequential File Processing

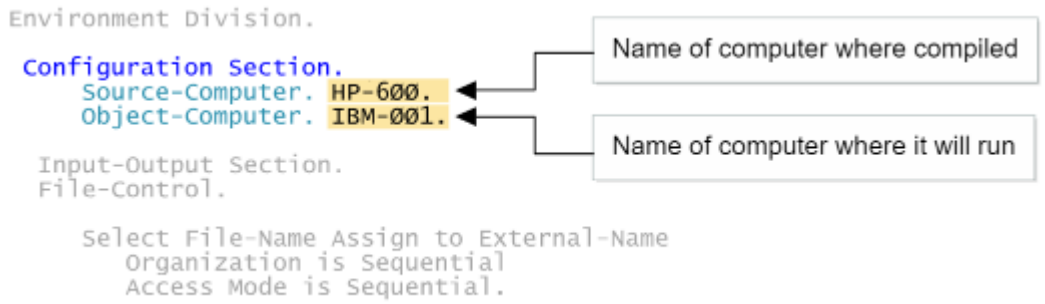
The Environment Division

Within the Environment Division, there is information in the Configuration and Input-Output sections that relate to sequential file processing.

Configuration Section

The Configuration Section is optional, and not used by many programs. Its purpose is for documenting various aspects of the program, including:

- The name of the computer where the program is compiled
- The name of the computer where the program will run
- Special items such as the currency sign and symbolic characters
- User-defined classes for object-oriented COBOL



Note: The source computer and object computer statements were important in earlier versions of COBOL where a program may have been compiled on one model of the manufacturer's hardware, but intended to run on a different model. For example, the machine code instruction set available on the source computer would not be available in its entirety on the object computer.

Input-Output Section

The Input-Output section includes definitions for all file operations, together with any other operations involving the transfer of data between a COBOL program and an external media. This can include output to printers, input from data entry or scanning devices, as well as file input and output.

File Control:

Holds definitions for every file accessed by the program and specifies details such as:

- External filename
- File organisation
- How it will be accessed

```
Environment Division.

Configuration Section.
    Source-Computer. HP-600.
    Object-Computer. IBM-001.

 Input-Output Section.
 File-Control.

    Select Input-File Assign to "my.file.name"
       Organization is Sequential
       Access Mode is Sequential.

    Select Output-File Assign to "my.output.name"
       Organization is Sequential
       Access Mode is Sequential.

 I-O-Control.
    Rerun On Input-File Every 50 Records.
```

I-O-Control:

This is an optional section specifying advanced I/O features for files defined in the File-Control area. These features can include check-pointing, sharing memory between files and the location of files stored on multiple tape units.

Select and Assign Entry:

Every file accessed by the program must have a separate Select entry in the File-Control area of the Input-Output section.

```
Environment Division.

Input-Output Section.
File-Control.

   Select Input-File Assign to "my.file.name"
      Organization is Sequential
      Access Mode is Sequential.
```

- Every file must have one and only one select statement.
- The ‘input-file’ specifies a name that the program will use when referencing the file. This can be any valid name in COBOL. Every select statement must use a different internal file name.
- The assign to statement states that the external name of the file, the name of the file used by the operating system, follows.
- The “my.file.name” is the name of the file used by the operating system; this can be one of:
    - Full filename or pathname
    - A file reference on mainframes this may be the DD name previously allocated to this file
    - A variable defined in the Data Division, holding the name of the file

- The organisation (optional) of the file for sequential files this can be:
    - Sequential - usually record sequential however some COBOL compilers include options that assume line sequential
    - Record Sequential
    - Line Sequential
- Other possible values are:
    - Indexed - for indexed files
    - Relative - for relative files
    - Record Binary Sequential - GNUCOBOL
    - Transaction - IBM i COBOL
- If omitted, organisation is sequential will be assumed.
- Access mode (optional) is how the file will be accessed and this must be sequential for all sequential files.

Select Options:

There are several different optional parameters and formats when using the Select statement for an input or output file. Not all COBOL compilers will support all of these options.

- Select Optional indicates that this file is optional; it may not exist or be provided every time the program runs
- Select Not Optional indicates that this file must be specified every time the program runs; this is the same as if Optional is not specified
- The File Status parameter can be used to specify a data area, usually defined in working storage, that will hold a numeric value at the conclusion of any I-O operation on the file. The number can be checked by the program to test if the operation was a success or not. Values also indicate other information such as EOF or reasons for failure.
- The Sharing With clause specifies how the file will be shared:
    - All other - access is shared with other processes
    - Read Only - other processes can read, but not write to this file
    - No Other - no other process can access this file
    - In many systems, the sharing is specified when the file is opened. If sharing is not specified, it usually defaults to read-only for input files and no other for output files.

Printed File:

A file may be assigned to a printer rather than a physical file. This allows a COBOL program to send output directly to a printer.

The name of the printer is usually assigned by the operating system, for example printer1.prn. On mainframe systems, the printer can be assigned a name in the JCL.

```
Environment Division.

Input-Output Section.
File-Control.

    Select Printed-Report Assign to Printer
       Organization is Sequential
       Access Mode is Sequential.
```

The Data Division

This Division is used to define the record content and format of the file. This can be done via the File Section.

File Section

Every file used by the COBOL program must have two definitions in the File section:

One File Descriptor Definition:

This defines the structure and options for the file. At its simplest, the File Descriptor specifies the name of the file that it relates to in the File-Control area of the Input-Output section. The file name in the file descriptor must match the file name in the corresponding Select statement.

- The File Descriptor definition has several options that define the processing or structure of the file. Not all options are supported by all COBOL compilers or on all systems.
- Recording Mode specifies if the records in the file are Fixed (F) or Variable (V) in length.
- The 'Block contains' specifies the number of records or characters to be stored in each block. On some systems such as IBM mainframes, this value can improve file processing performance.
- The 'Data Record' specifies the record descriptor describing the record format. In modern systems, this is no longer required, and the record descriptors immediately following the file descriptor are used.
- The 'External' specifies that this is an external file descriptor. This means that other programs in the same process or run-unit can also use this file. This could be used for program-to-program communications. There can only be one external file per process or run-unit.
- The 'Global' specifies that this is a global file descriptor and any subprogram can also access this file.
- The 'Linage' defines a logical page. The number of lines per page, top margin (Lines at Top), bottom margin (Lines at Bottom), and footer (Lines with Footing) are specified. It is only used for output files.

One Record Descriptor Definition:

This defines the format of the file's records, and must immediately follow the File Descriptor record.

- The Record Descriptor defines the format of the records for the file. At its simplest, it is the same format as Working-Storage definitions, using Picture definitions to define formats.
- It must immediately follow its corresponding file-descriptor.
- As with working-storage, record descriptors can have multiple levels and data types.
- A file descriptor can have more than one record descriptor. This would be used if a file has more than one possible record format.

Sequential File Processing Modes

Once all required definitions have been completed in the Environment and Data Sections, the file is ready for use. The COBOL program can access the file during normal processing in the Procedure Division.

A COBOL program can perform four operations on sequential files Open, Read, Write, and Close.

Syntax - Open Statement:

Before a file can be accessed, it must be opened. Open processing requests the operating system to prepare a file for use. The operating system will check security access, allocate storage and other resources, perform any locking required, and other processing needed before a file can be used.

- Alternatives to using input include:
    - Output - new file will be created, or an existing one replaced
    - Extend - the file already exists and new records will be added to the end of it
    - I/O - a file can be read or written and only for record sequential files on disk
- Multiple opening statements can be grouped together

Syntax - Read Statement:

A Read statement on a sequential file does exactly that, it reads the next record in the file. When a sequential file is opened, it points to the first record. After each read, this pointer is moved to the next.

If a program needs to read a record before the current record pointer, the file will need to be closed, and re-opened.

- The optional read FILEINPUT into RECORD can be used to move the record into a data area defined in the working-storage section.
- If there are no more records to read, no data will be read. This can be tested by using the At End clause. This specifies processing to be performed when the end of the file is reached.
- The End-Read statement is normally used to specify the end of statements to process at the end of a file.
- The Not At End clause specifies processing to be performed if a record was successfully read, and not at EOF.
- Sequential file reads will usually be specified inside a perform statement.

Syntax - Write Statement:

A Write statement on a sequential file writes one record to the end of the sequential file.

- The optional write FILEOUT from RECORD can be used to move data from a data area defined in the working-storage section.

Concept - Write to Printer:

The Write statement can also be used to send output to a printer, and includes optional clauses to control the output printed.

```
Procedure Division.
Open Output Output-File

Write Output-Printer-Record After Advancing 1 Lines

Write Output-Printer-Record Before Advancing WS-Num Lines

Write Output-Printer-Record After Advancing Page

Write Output-Printer-Record Before Advancing Page

Write Output-Printer-Record
   At End-of-Page Display "End of Pager"
End-Write
```

Syntax - Close Statement:

When a program has finished with a file, it must be closed. This instructs the operating system to release any resources it needs to access the file, and allow other processes access. Normally files are automatically closed when a process or run-unit finishes. However it is good practice to specifically close all files opened in a program when they are no longer needed.

Programs sometimes will close a file, then re-open it. This could be because:

- A program has written a file, and now wants to read it
- A program needs to re-read a sequential file
- A program has read a file, and will now replace the entire file

The program may re-open the file for the same operation, Read, Output, Extend, I/O, or for a different operation.