

Table of Contents:

Table of Contents:	1
File Processing	4
Hierarchy of Data	4
Introduction	4
Data Structure	4
Cobol's Basic Elements	5
Defining Data Structure	5
File Concepts	6
Master Files	6
Example:	6
Example:	6
Transaction Files	6
Example:	6
Reference Files	6
Example:	6
Access to Files	7
Serial:	7
Sequential:	7
Direct:	7
Indexed Sequential:	8
Relative/Random Files:	8
File Operations	8
Syntax - Open:	8
Syntax - Close:	9
Syntax - Read:	9
Syntax - Write:	9
Concept - Buffering:	9
Sequential File Processing	10
The Environment Division	10
Configuration Section	10
Input-Output Section	10
File Control:	10
I-O-Control:	11
Select and Assign Entry:	11
Select Options:	12
Printed File:	12
The Data Division	13
File Section	13
One File Descriptor Definition:	13
One Record Descriptor Definition:	13
Sequential File Processing Modes	14

Syntax - Open Statement:	14
Syntax - Read Statement:	14
Syntax - Write Statement:	14
Concept - Write to Printer:	15
Syntax - Close Statement:	15
Direct File Processing - Indexed Files	16
Direct Access Files - Indexed	16
Indexed Files:	16
Processing Indexed Files	17
Sequential Processing:	17
Direct Processing:	17
Skip-Sequential Processing:	18
Index Structure	18
COBOL Definition	19
Organisation Clause:	19
Access Mode Clause:	19
Record Key Clause	19
Alternative Record Key Clause	20
Record Structure:	20
Index File Processing Modes	20
Read Modes	20
Sequential Read:	21
Skip-Sequential Read:	21
Direct Read:	22
Direct Read - Alternate Index:	22
Index File Statements	23
Syntax - Start Statement:	23
Relational Operators:	23
Syntax - Sequential Delete:	24
Syntax - Direct Delete:	24
Syntax - Write Statement:	25
Concept - Write Statement Failure:	25
Syntax - Rewrite Statement:	26
Direct File Processing - Relative Files	27
Process Relative Files	28
Calculating Record Number	28
Processing Relative Files	28
Sequential Processing:	29
Direct Processing:	29
Skip-Sequential Processing:	29
Implement Relative Files	30
COBOL Definition	30
Organisation Clause:	31

Access Mode Clause:	31
Relative Key Clause:	31
Record Structure:	31
Calculate the Record Number:	32
Calculate the Record Number - Alt:	32
Read Relative Files	33
Sequential Read:	34
Syntax - Start Statement:	34
Direct Read:	35
Syntax - Write Statement:	35
Syntax - Rewrite Statement:	36
Syntax - Sequential Delete:	36
Syntax - Direct Delete:	37

File Processing

Hierarchy of Data

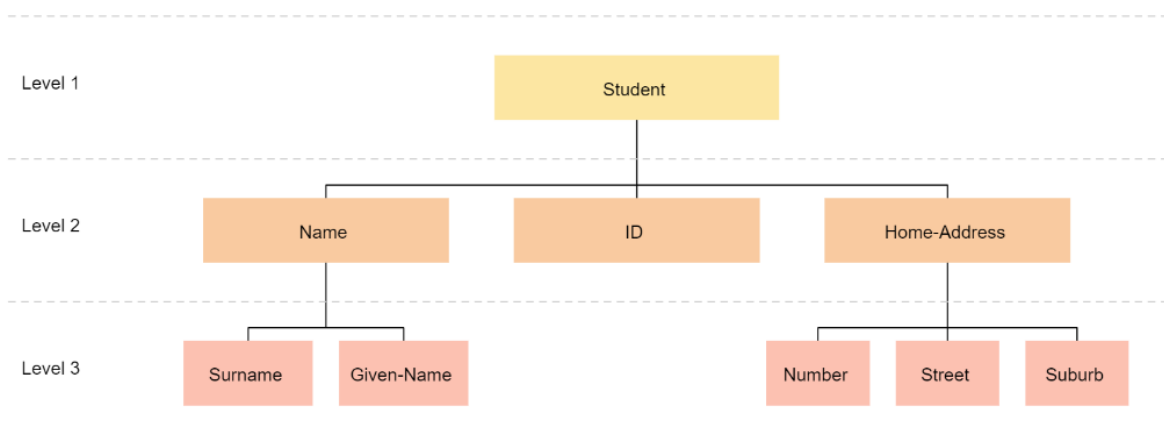
Introduction

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

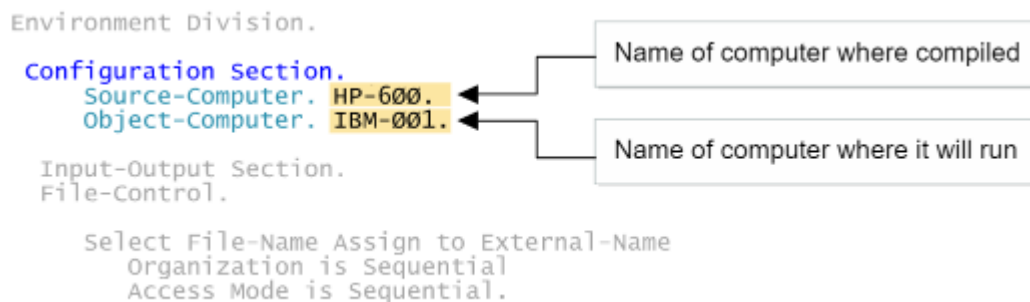
The Environment Division

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.

Direct File Processing - Indexed Files

In the previous module you explored how COBOL programs define and access sequential files. COBOL programs can also access indexed files, either sequentially, or directly by locating individual records using the index.

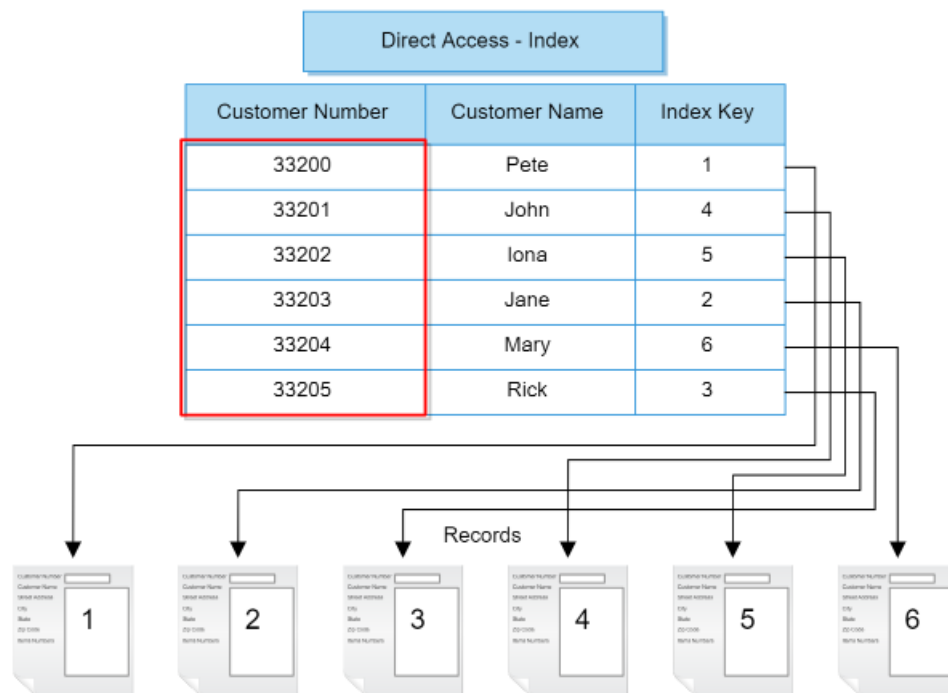
Direct Access Files - Indexed

There are two basic methods of providing direct access to records. The first method locates records by the use of an index, the second method generates a file address for each record by a mathematical formula applied to its key field.

Indexed Files:

Indexed files, often referred to as indexed sequential files, provide direct access by the use of one or more indexes, each of which relates to a key field in the records.

Note, the records in an indexed file need not be stored in key order, this depends on how the file is physically defined.



The main index is referred to as the primary index. This primary index must be unique: every record must have a different value for the primary index.

For example, a customer database may have a field holding a number uniquely identifying each customer. This field would be perfect as a primary index.

Other fields can also be used as indexes. Indexes other than the primary index are called secondary indexes, or alternate indexes. Alternate indexes do not have to be unique - more than one record can have the same value.

For example, our customer database may use the customer's name as an index: a customer's record could then be quickly found by searching on the customer's name. There may be two customers with the same name.

Processing Indexed Files

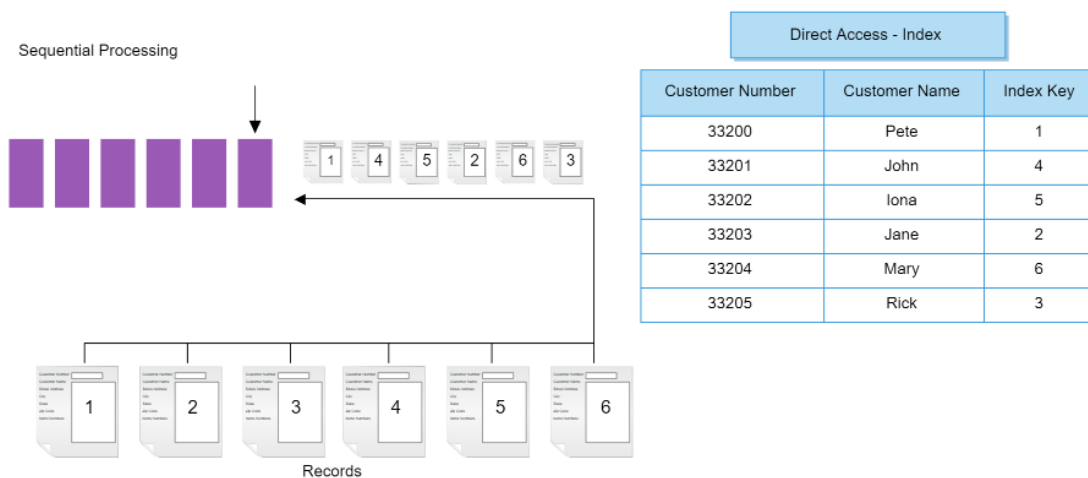
Indexed files can be processed in three different ways:

- Sequential processing
- Direct processing
- Skip-Sequential processing



Sequential Processing:

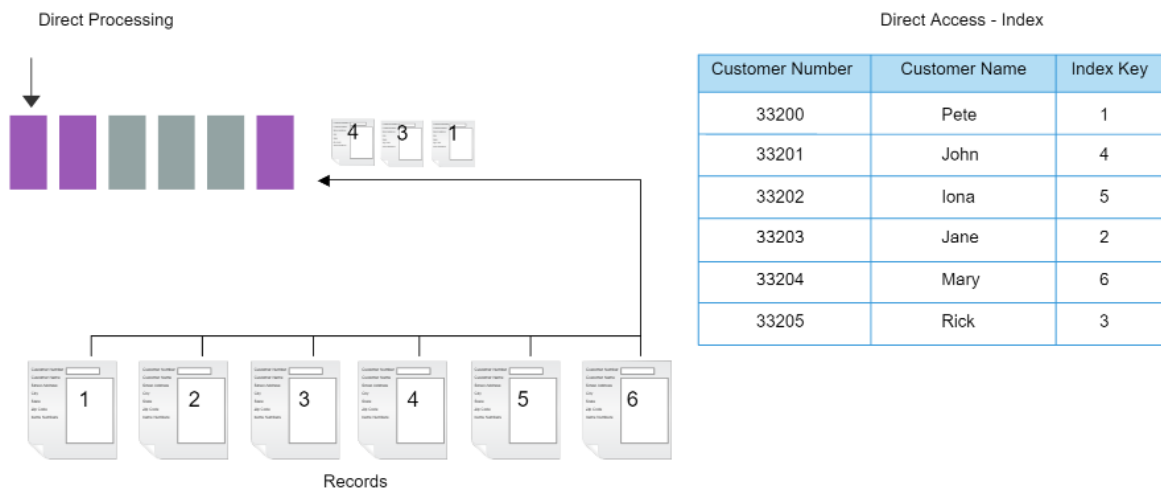
Indexed files can be processed sequentially, in which case, records will be retrieved in key sequence.



Direct Processing:

Indexed files can also be processed directly. In this case the program specifies a value for one of the indexes: a key. By specifying this key, the program can immediately locate the record required. Records can be obtained in any order.

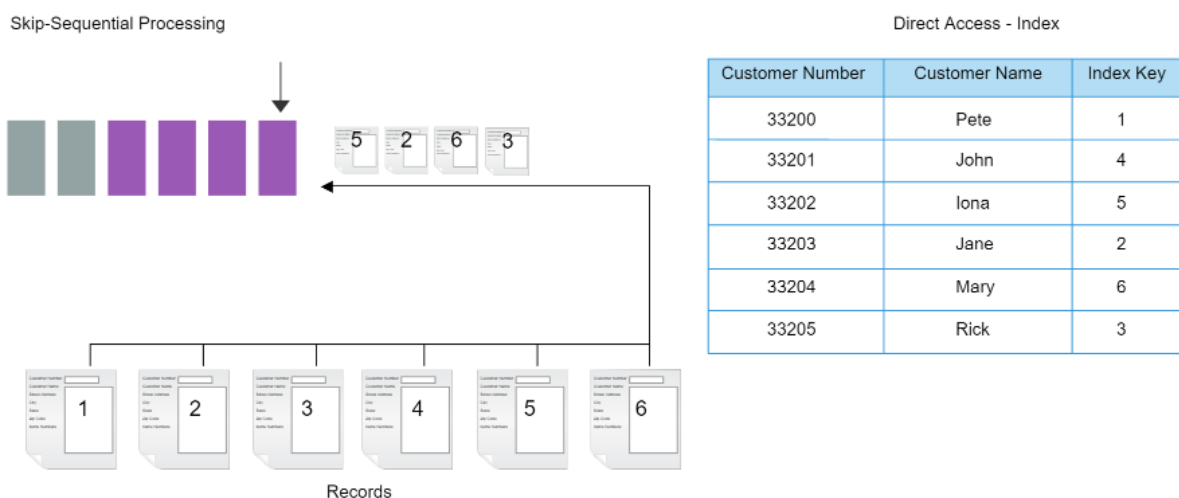
In the image below it goes from 2nd pos to last post to first pos hence, 4 - 3 - 1.



Skip-Sequential Processing:

In addition, indexed files can be processed by a mixture of sequential and direct methods. In this mode, a program can directly access a particular record and then read sequentially from that point onwards.

For example, COBOL might access the first employee with a unique staff ID number of 33202 and then read all employee ID numbers that are greater including, 33203, 33204, and 33205.



Index Structure

Different systems will implement indexed files in different ways. The file may be stored differently, and have different index structures. For example, some store an index entry for every key value in the file, others arrange records in blocks, and store an index for the first value in the block. Some use a list structure for the index, others a tree structure.

Regardless of how indexed files are implemented by the operating system, the COBOL statements and processing are the same. COBOL programmers **do not need to know** how the operating system implements indexes and indexed files.

COBOL Definition

Like sequential files, every indexed file used by a COBOL program must have one definition in the File-Control area of the Input-Output Section in the Environment division.

```
Environment Division.  
Input-Output Section.  
File-Control.  
  
    Select Personnel-File Assign to Staff  
        Organization is Indexed  
        Access Mode is Random  
        Record Key is Staff-ID-Number  
        Alternate Record Key is Employee-Surname With Duplicates  
        Alternate Record Key is Department-Code With Duplicates.
```

Organisation Clause:

Indexed files must always be defined with the clause Organization is Indexed.

Access Mode Clause:

The Access Mode clause specifies how the file will be processed. This value can be one of the following:

- Sequential
 - The Access Mode is Sequential clause specifies that the file will be accessed sequentially. This clause prevents a program from accessing the file in any other way, no direct or skip-sequential access.
- Random
 - The Access Mode is Random clause specifies that the file will be accessed directly or randomly. Every access must specify a key, sequential or skip-sequential access is not possible.
 - This could be used to enhance security, limiting a program to the records it knows about, to the records it has the key for.
- Dynamic
 - The Access Mode is Dynamic clause specifies that the file can be accessed sequentially or randomly.
 - This is required if performing skip sequential processing, or another combination of direct and sequential access.

Record Key Clause

The Record Key clause specifies the data area name that is the primary key for the file. The COBOL program can use this field to locate a record using the file's index.

- This data area must be defined in the record descriptor record in the File section of the Data Division.
- Remember that this is the primary key, so the value must be unique in the file. Two records cannot have the same value for this data area.

Alternative Record Key Clause

The Alternate Record Key clause is optional and, if included, it may be used a number of times in the same Select statement. It is used to describe one or more fields that are secondary indexes for the file.

Like the primary key, the data items referred to as the alternate keys must also be fields within the record defined for the file.

- The With Duplicates specification is optional for alternate keys.
- If it is not specified, the alternate keys must be unique, two records cannot have the same value for that field. It is more common to allow for duplicate values in alternate keys..

Record Structure:

The record structure for any file must be defined in the FD entry in the File-Section of the Data Division.

For indexed files, this definition must include the primary key field, and any secondary key fields referred to in the Select statement in the Input-Output section.

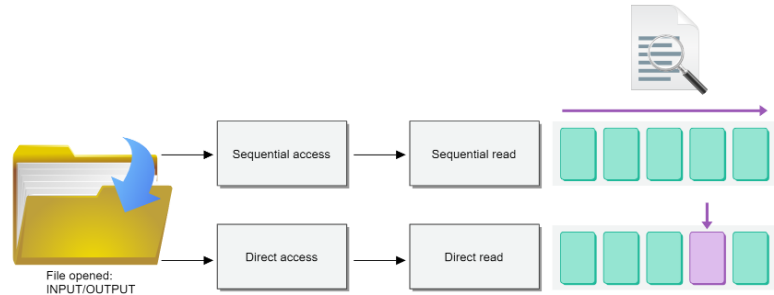
Index File Processing Modes

As with sequential files, there are four basic operations that are performed on indexed files, Open, Read, Write, and Close.

- As with all files in a COBOL program, an indexed file must be opened before it can be accessed. You do this the same way you would a sequential file
- When a program has finished with any file, including an indexed 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.

Read Modes

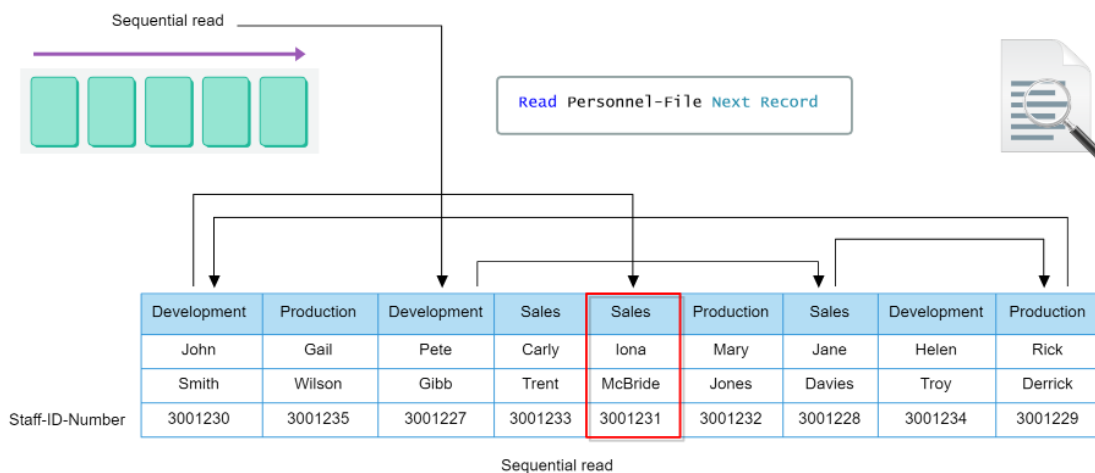
Indexed files allow sequential and direct access to their records. The two corresponding reading modes are Sequential read and Direct read. The file must be opened in Input or I-O mode for these reads to be possible.



Sequential Read:

This example shows an indexed file named "Personnel-File" that contains employee records. The key for each record is the Staff-ID-Number.

A sequential read on the Personnel file will present the program with the next record in key or Staff-ID-Number sequence.



To enable a sequential read, the access mode selected in the description of the file in the Environment Division must be either Sequential or Dynamic.

As with sequential files the Read statement is used to perform a sequential read of an indexed file.

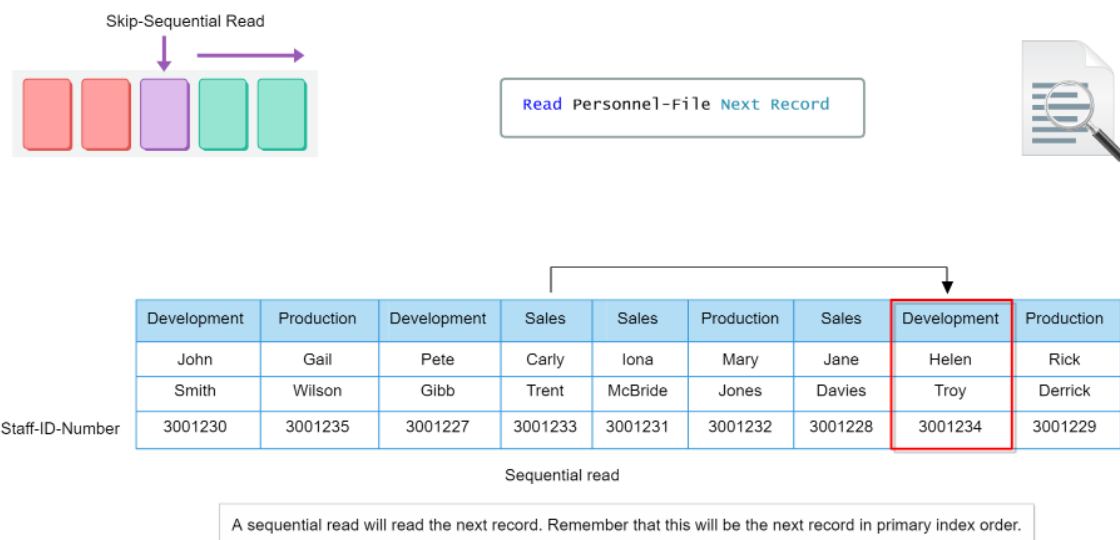
Procedure Division.
 Open Input Input-File
 Read Input-File Next Record

The Next Record phrase instructs COBOL to perform a sequential read. It is not required if the access mode is sequential, only sequential reads are permitted in this case.

Skip-Sequential Read:

A sequential read will read the next record, in key index order. This would normally start at the first record.

However records can also be directly accessed in an indexed file. If a record is first directly accessed, sequential reads will then sequentially process records from this first record. This is skip-sequential processing.



Direct Read:

A direct read attempts to retrieve a specific record from a file. A program identifies this record by specifying the key of the record.

A direct read can only be performed on a file that has been opened for read or I/O, and an access mode of random or dynamic.

- In a direct read, the key of the record to be read must first be placed in the primary key field defined in the file descriptor.
- If a record with a specified key cannot be found, no record can be retrieved. Processing to be performed when such an 'invalid key' error occurs can be specified in the **Invalid Key** phrase of the read statement.
- Similarly you can use the **Not Invalid Key** phrase for the opposite scenario.

Data Division.
File-Section.

```
FD Personnel-File.  
01 Employee-Record.  
02 Staff-ID-Number          Pic 9(6).  
02 Employee-Name.  
03 Employee-First-Name     Pic X(20).  
03 Employee-Surname        Pic X(20).  
02 Department-Code         Pic X(20).
```

```
Procedure Division.  
Open Input Personnel-File  
Move "300123" to Staff-ID-Number  
Read Personnel-File  
Not Invalid Key  
Display "Record Successfully Retrieved"  
End-Read
```

Direct Read - Alternate Index:

Records may also be read using alternate keys defined for a file.

- A read statement by default will use the primary key to locate the desired record, in this case, Staff-ID-number.
- To locate a record using an alternate key, the **Key Is** phrase is used in the read statement. In this example, the alternate index Department-Code is used.
- A value for this alternate index must first be moved into the relevant field in the record descriptor.
- As with a read using the primary key, processing when a record is, or is not, read can be specified using the invalid key and not invalid key phrases.

```
Data Division.
File-Section.

FD Personnel-File.
01 Employee-Record.
02 Staff-ID-Number          Pic 9(6).
02 Employee-Name.
03 Employee-First-Name     Pic X(20).
03 Employee-Surname        Pic X(20).
02 Department-Code         Pic X(20).

Procedure Division.
Open Input Personnel-File
Move "Marketing-001" to Department-Code
Read Personnel-File Key is Department-Code
Invalid Key
    Display "Record Not Found"
Not Invalid Key
    Display "Record Successfully Retrieved"
End-Read
```



Index File Statements

COBOL also provides features to locate, delete, insert, and update records in an indexed file.

Syntax - Start Statement:

In situations where you need to locate a record, the Start statement can be used to position the file pointer on that record, ready for sequential processing. This is how skip-sequential processing is performed. Start can only be used in a file that has been opened in Sequential or Dynamic mode.

The Start statement does not read a record. It only locates a file, from which skip-sequential processing can continue.

If the Key Is phrase is not specified, the first record matching the primary key is performed.

```
Data Division.
File-Section.

FD Personnel-File.
01 Employee-Record.
02 Staff-ID-Number          Pic 9(6).
02 Employee-Name.
03 Employee-First-Name     Pic X(20).
03 Employee-Surname        Pic X(20).
02 Department-Code         Pic X(20).

Procedure Division.
Open Input Personnel-File
Move "300123" to Staff-ID-Number
Start Personnel-File
Key is = Staff-ID-Number
Not Invalid Key Display "Key found"
End-Start
```



Relational Operators:

This is the full list of COBOL standard relational operators that can be implemented in programs. Most COBOL programmers use the "Is Not Less Than" operator for Start/Read processing and "Is Equal To" for indexed Read processing.

Simple Operators	Extended Operators
Is Greater Than	Is Greater Than Or Equal To
Is >	Is >=
Is Less Than	Is Not Less Than
Is <	Is Not <
Is Equal To	Is Less Than Or Equal To
Is =	Is <=
	Is Not Greater Than
	Is Not >
	Is Not Equal To
	Is Not =
	Is <>

Syntax - Sequential Delete:

Records can be deleted using the Delete statement. How the Delete statement works is different for sequential and random or dynamic access modes. The Delete statement cannot be used for a file that has been specified as Organization is Sequential.

- The success or failure of a Delete statement can be performed by checking the variable defined in the file status phrase of the select statement.

```
Environment Division.

Input-Output Section.
File-Control.

    Select Personnel-File Assign to Staff
        Organization is Indexed
        Access Mode is Sequential
        Record Key is Staff-ID-Number
        File Status Status-Area.

Data Division.
File Section.

FD Personnel-File.
01 Employee-Record.
02 Staff-ID-Number          Pic 9(6).
02 Employee-Name.
03 Employee-First-Name     Pic X(20).
03 Employee-Surname        Pic X(20).
02 Department-Code         Pic X(20).

Procedure Division.
Open I-O Personnel-File
Read Personnel-File

Delete Personnel-File Record
If (Status-Area = 0)
    Display "Record Deleted"
End-If
```


Syntax - Direct Delete:

The Delete statement is also used to delete records when accessing an indexed file directly. In this case, the key of the record to be deleted is specified.

- Unlike sequential access, it is not necessary to read a record before deleting it.
- The primary key of the file to be deleted is moved to the record descriptor (Alternate keys cannot be used).
- If a record with the specified key is not found, no record is delete

```
Environment Division.

Input-Output Section.
File-Control.

    Select Personnel-File Assign to Staff
    Organization is Indexed
    Access Mode is Random
    Record Key is Staff-ID-Number.

Data Division.
File Section.

FD Personnel-File.
01 Employee-Record.
   02 Staff-ID-Number           Pic 9(6).
   02 Employee-Name.
      03 Employee-First-Name    Pic X(20).
      03 Employee-Surname       Pic X(20).
   02 Department-Code          Pic X(20).

Procedure Division.
Open I-O Personnel-File
Move "300123" to Staff-ID-Number
Delete Personnel-File Record
    Not Invalid Key Perform Report-Employee-Deleted
End-Delete
```

Syntax - Write Statement:

There are two circumstances in which a record needs to be written to an indexed file. These are when placing a new record on the file and when overwriting an existing record. The Write statement is used in COBOL to insert a new record into an indexed file.

- If the access mode is sequential, the file may also be opened in Extend mode on most COBOL compilers. In this case, new records are added to the end of the file
- The record to be inserted must be moved into the record descriptor before using the write statement.
- The access mode is normally random or direct. This allows a record to be inserted anywhere in the file.
 - However, records can also be inserted in sequential access mode. Different compilers may have different rules as to how this can occur.
- If a record already exists with the same primary index, the record will not be inserted

```
Environment Division.

Input-Output Section.
File-Control.

    Select Personnel-File Assign to Staff
    Organization is Indexed
    Access Mode is Random
    Record Key is Staff-ID-Number.

Data Division.
File Section.

FD Personnel-File.
01 Employee-Record.
   02 Staff-ID-Number           Pic 9(6).
   02 Employee-Name.
      03 Employee-First-Name    Pic X(20).
      03 Employee-Surname       Pic X(20).
   02 Department-Code          Pic X(20).

Procedure Division.
Open I-O Personnel-File
Move New-Record to Employee-Record
Write Employee-Record
    Invalid Key Display "Duplicate Record"
End Write
```

Concept - Write Statement Failure:

It is possible to declare alternate keys to allow, or not allow duplicates. Primary keys must always be unique. Similarly alternate keys that do not allow duplicates must also be unique in the file. The Invalid Key phrase of the Write statement can be used to check for a duplicate Key Error.

```
Environment Division.

Input-Output Section.
File-Control.

    Select Personnel-File Assign to Staff
    Organization is Indexed
    Access Mode is Random
    Record Key is Staff-ID-Number
    Alternate Record Key is Employee-Surname.

Data Division.
File Section.

FD Personnel-File.
01 Employee-Record.
02 Staff-ID-Number          Pic 9(6).
02 Employee-Name.
03 Employee-First-Name     Pic X(20).
03 Employee-Surname        Pic X(20).
02 Department-Code         Pic X(20).

Procedure Division.
Open I-O Personnel-File
Move New-Surname to Employee-Surname

Write Employee-Record
Invalid Key Perform Report-Duplicate-Key
End-write
```



Syntax - Rewrite Statement:

The Rewrite statement is used in COBOL to update an existing record in an indexed file. The statement must specify the record descriptor of the file, not the file descriptor.

- The file can be opened in random, dynamic or sequential access mode.
- If the access mode is sequential, a record must first be read. This record is then updated by the rewrite.

```
Procedure Division.
Open I-O Personnel-File

Read Personnel-File
Move New-Department-Code to Department-Code
Rewrite Employee-Record
```

- If directly accessing the file, a record can also be read, updated and re-written.

```
Procedure Division.
Open I-O Personnel-File
Move "300133" to Staff-ID-Number
Read Personnel-File
Move New-Department-Code to Department-Code
Rewrite Employee-Record
```

- Using direct access, the primary key and the new values may be moved into the record descriptor without a prior read. This is not often done, however, most programs will first read the record to be updated.

```

Procedure Division.
Open I-O Personnel-File
Move "300133" to Staff-ID-Number
Read Personnel-File
Move Required-Staff-ID-Number to Staff-ID-Number
Move New-Department-Code to Department-Code
Move New-Employee-First-Name to Employee-First-Name
Move New-Employee-Surname to Employee-Surname
Rewrite Employee-Record

```





- Note that the primary key cannot be changed using a rewrite statement. To change the primary key, the record must first be deleted, and then recreated using the write statement.



COBOL



File

Procedure Division		
	Locate	Open I-O File-Descriptor Move Target-Key to Pri-Key Start File-Descriptor
	Delete	Open I-O File-Descriptor Move Target-Key to Pri-Key Delete File-Descriptor Record
	Insert	Open I-O Personnel-File Move New-Record to Record Write Record-Descriptor
	Update	Open I-O Personnel-File Read File-Descriptor Move New-Field to Field Rewrite Record-Descriptor

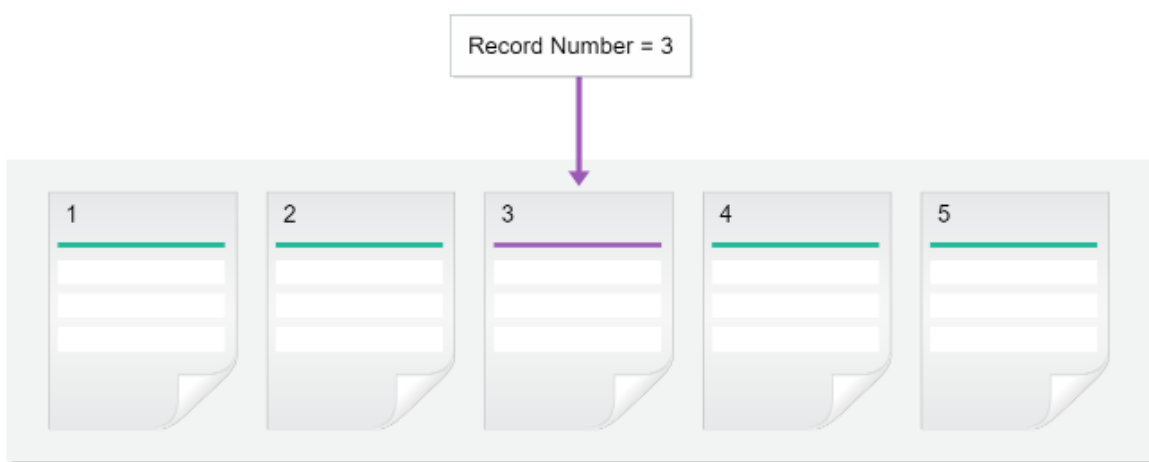
Direct File Processing - Relative Files

As seen above, a record in an indexed file can be accessed directly by specifying a key.

Relative files can also be directly accessed. However a record is accessed by specifying the record number. For example, to access the first record, a program would specify 1, for the third record, 3.

Process Relative Files

Direct access to records in a relative file is usually faster than an indexed file, as no time is required to search and update indexes. Relative files also use less disk space than indexed files.



Calculating Record Number

The difficulty when working with relative files is for a program to know the number of the record required. How can a program know that it needs record number 3?

Programs often use a formula or record number generating algorithm to calculate the record number. This formula is applied to one or more fields of a record, and returns a number corresponding to the record number in the file.

In this example, every staff member has a unique Staff ID, starting at 1. This ID is also used as the record number in the file.

Processing Relative Files

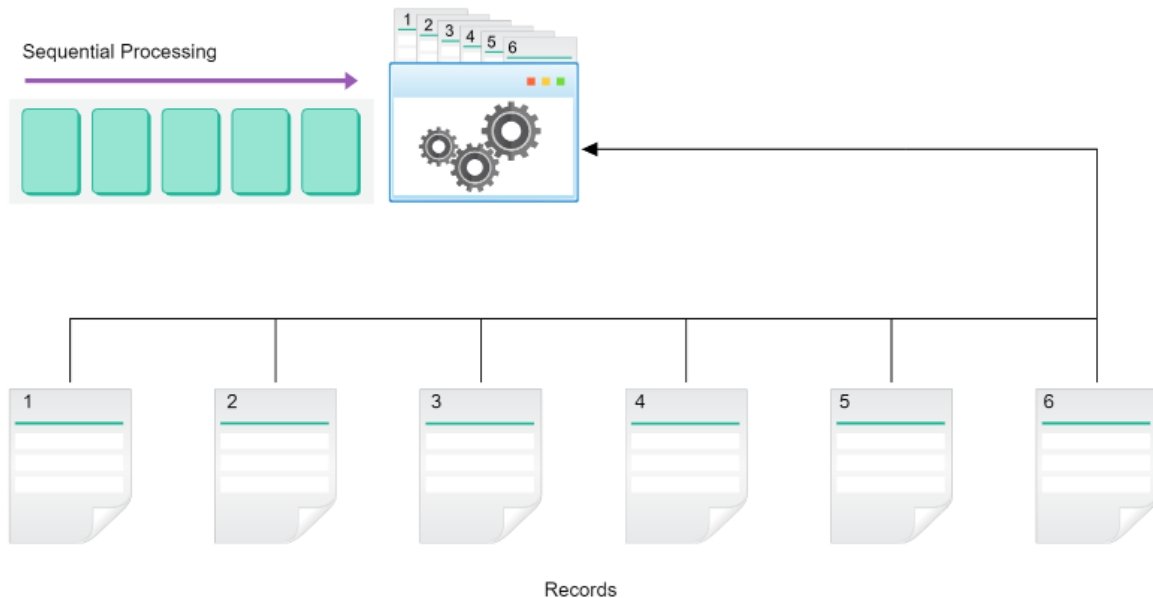
There are three methods of processing relative files are:

- Sequential processing
- Direct processing
- Skip sequential processing

Sequential Processing:

Relative files can also be processed sequentially. In this case, records will be retrieved in the sequence in which they are held on the file. This sequence will be a physical sequence rather than a key sequence.

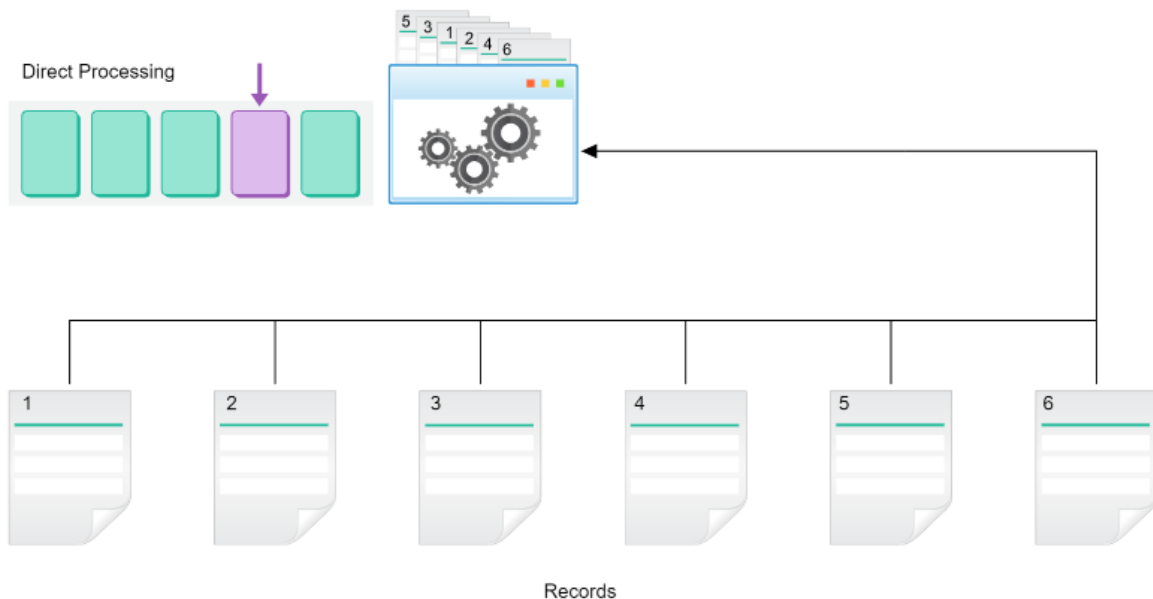
- Whether or not this physical sequence corresponds with the key sequence depends on the address-generating algorithm.



Direct Processing:

Relative files can be processed directly. In this case, the program specifies the record number, and this record is accessed immediately.

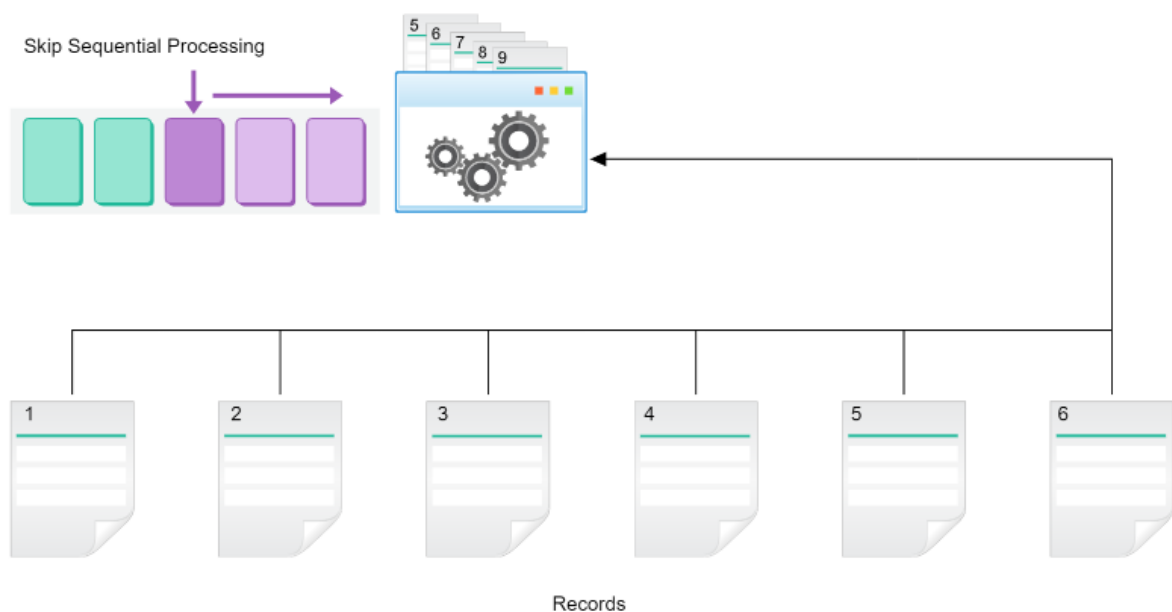
- For the record to be processed, the program must calculate or obtain the record number.



Skip-Sequential Processing:

Relative files can also be processed by a mixture of sequential and direct methods. In this mode, a program can directly access a particular record and then read sequentially from that point onwards. This is called Skip Sequential processing.

- For example, it may access the first record with a position number greater than 1000 and then read all the following records until the end of the file or some terminating point is reached.
- It is important to note that some data areas in a relative file may be empty, containing no records.
 - When you read a relative file sequentially, the program will automatically skip these areas. If you attempt to access an empty area directly, you will get an invalid key.



Implement Relative Files

COBOL Definition

Like sequential and indexed files, every relative file used by a COBOL program must have one Select definition in the File-Control area of the Input-Output section in the Environment division.

```
Environment Division.  
  
Input-Output Section.  
File-Control.  
  
    Select Statistics-File Assign to Statistics  
        Organization is Relative  
        Access Mode is Random  
        Relative Key is Stats-Record-Number.
```

Organisation Clause:

The first difference you will see is that relative files must always be defined with the clause Organization is Relative.

Access Mode Clause:

As with indexed files, the Access Mode clause specifies how the file will be processed. This value can be one of:

- Sequential
 - The clause Access Mode is Sequential restricts access to the records to sequential processing. A program can not directly access a record by specifying the record number.
 - This can be used in cases where every record in the file must be processed in the sequence in which it is stored and direct access is therefore inappropriate.
- Random
 - The clause Access Mode is Random restricts the processing of the records to random access only, a program must specify the record number. Sequential processing is not permitted.
 - Such a method would prevent a program from accessing records for which it did not know the record number. This might provide a measure of security from unwarranted access to records that the program does not need.
- Dynamic
 - The clause Access Mode is Dynamic allows the records on the file to be processed with either sequential access or random access as required by the program from time to time.
 - It is a combination of the sequential and random access modes.

Relative Key Clause:

The Relative Key clause specifies the data area that hold the record number of the file to be processed. If record number 4 is to be directly accessed, the number four is stored in this data area, which must be defined in the Working-Storage Section. This is how a specific record is accessed.

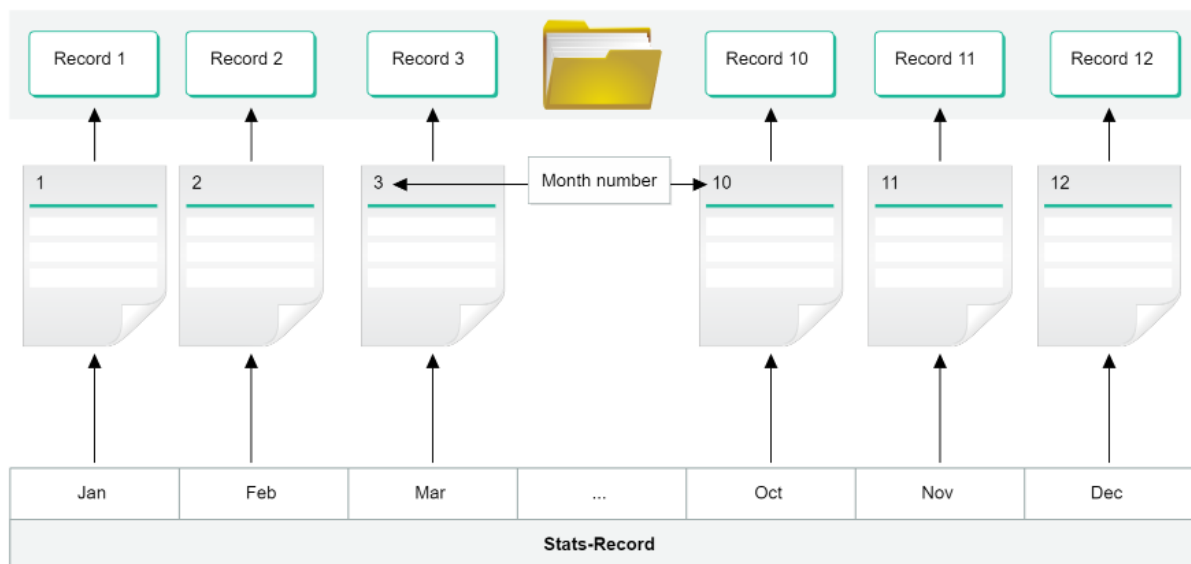
Record Structure:

Each relative file has a file-descriptor and record descriptor definition, as with sequential and indexed files. Remember that the record descriptor generally must include one or more fields that can be used to calculate the record number. In this example, the Month-Number field will be used as the record number.

```
File Section.  
FD Statistics-File.  
01 Stats-Record.  
02 Month-Number Pic 9(2).  
02 Salary-Details-This-Month.  
03 Monthly-Total-Gross Pic 9(8)V9(2).  
03 Monthly-Total-Tax Pic 9(8)V9(2).
```

Calculate the Record Number:

Take for example a file like the Statistics-File shown here. In this file there are twelve records, one for each month.



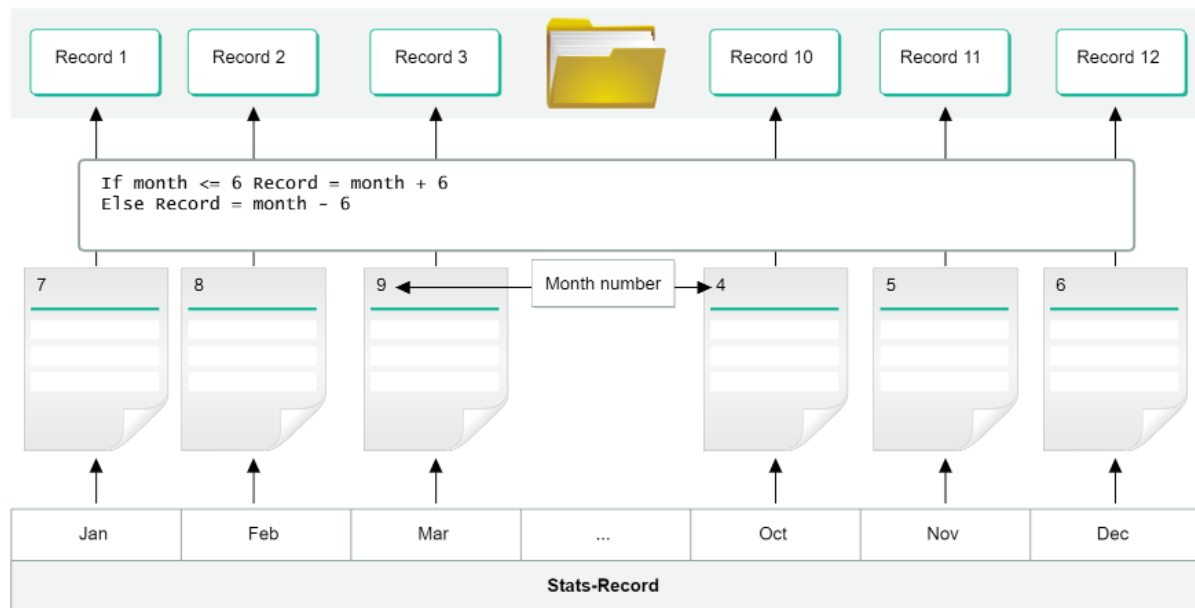
Each record holds totals relating to the payroll for one calendar month. The records can be stored in normal calendar sequence. In this case, there would be a simple relationship between the:

- Month number
- Position of the record in the file
- The record for January contains a Month Number of 01 and is held in record position 1 in the file.
- The record for December contains a Month Number of 12 and is held in record position 12 in the file.

For example, to retrieve the record for August, the program would place a value of 08 in the Relative Key, Stats-Record-Number, and execute a Read instruction.

Calculate the Record Number - Alt:

Alternatively, a company may have a yearly reporting period from July to June. In this case, the records on the file may be stored with July (month number 7) as the first month, through to June (month number 6) as the 12th or last month.



In this case, the program would need to perform a calculation on the calendar number of any month to convert it to the Stats-Record-Number required as the Relative Key field for its retrieval.

Here, August, Month Number = 08, would need to be retrieved by means of a Relative Key of 02.

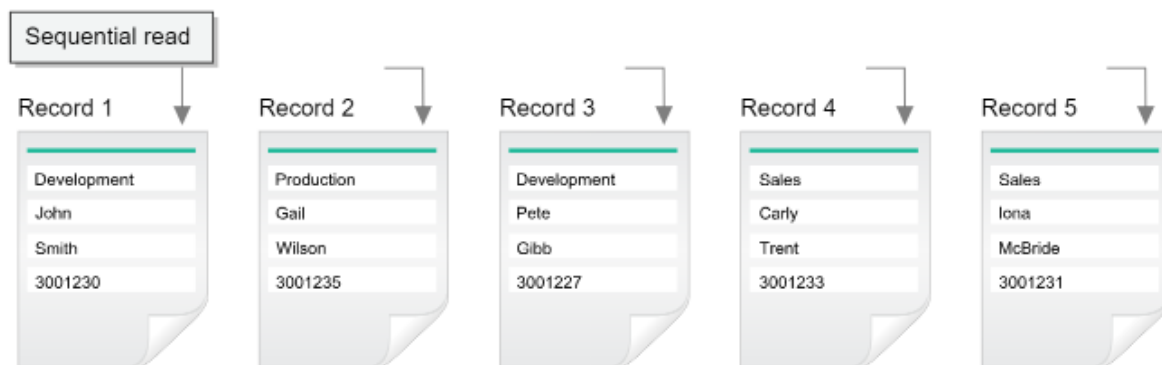
Read Relative Files

Working with relative files has many similarities to sequential and indexed files. As with these, there are four basic operations that are performed on relative files, Open, Read, Write, and Close. As with all files in a COBOL program, a relative file must be opened before it can be accessed.

As with indexed files, relative files can be read either sequentially or directly.

Sequential Read:

A sequential read of a relative file will access each record in record number order.



The syntax for a sequential read of a relative file is the same as for an indexed file, with the same phrases for testing for the end of the file. All sequential reads should test for the end of file as shown here, with processing to perform when this occurs.

```
Read File-Name Next Record
  At End Display "End of File"
  Not At End Display "Record Read"
End-Read
```

Syntax - Start Statement:

As with indexed files, the Start command can be used to locate a record. This can be used to position the file pointer to a specific record, ready for sequential processing. This is how skip-sequential processing is performed.

The Start command does not read a record, it only locates to a record in the file, from which skip-sequential processing can continue.

```
Data Division.
File Section.
FD Statistics-File.
01 Stats-Record.
02 Month-Number Pic 9(2).
02 Salary-Details-This-Month.
03 Monthly-Total-Gross Pic 9(8)V9(2).
03 Monthly-Total-Tax Pic 9(8)V9(2).

Working-Storage Section.
01 Stats-Record-Number Pic 9(5).

Procedure Division.
Open Input Statistics-File
Move 3 to Stats-Record-Number
Start Statistics-File
Key is = Stats-Record-Number
Invalid Key Display "Record not found"
End-Start
```

- The number for the record to locate must be placed in the correct relative key field specified in the File-Control section, Stats-Record-Number in this example, before the Start statement.
- The Key Is phrase specifies the comparison to be performed.
- You can also use Invalid Key/Not Invalid Key phrase here

Direct Read:

A direct read attempts to retrieve a specific record from a file. A program identifies this record by specifying the number of the record in the relative index defined in the File-Control section.

A direct read can only be performed on a file that has been opened for Input or I/O, and an access mode of Random or Dynamic.

```
Environment Division.

Input-Output Section.
File-Control.

Select Statistics-File Assign to Statistics
Organization is Relative
Access Mode is Random
Relative Key is Stats-Record-Number.

Data Division.
File Section.
FD Statistics-File.
01 Stats-Record.
02 Month-Number Pic 9(2).
02 Salary-Details-This-Month.
03 Monthly-Total-Gross Pic 9(8)V9(2).
03 Monthly-Total-Tax Pic 9(8)V9(2).

Working-Storage Section.
01 Stats-Record-Number Pic 9(5).

Procedure Division.
Open Input Statistics-File
Move 3 to Stats-Record-Number
Read Statistics-File
Invalid Key Display "Record not found"
End-Read
```

Syntax - Write Statement:

The Write command is used in COBOL to insert a new record into a relative file.

- The record to be inserted must be moved into the Record Descriptor before using the Write statement.
- The file must first be opened for I-O before using the write statement.
- If the access mode is random or dynamic, a direct write is requested. The record number where this new record will be inserted must be moved into the data area identified by the Relative Key clause.

```
Procedure Division.  
Open I-O Statistics-File  
Move 3 to Stats-Record-Number, Month-Number  
Move 12.50 to Monthly-Total-Gross  
Multiply Monthly-Total-Gross by 0.25  
Giving Monthly-Total-Tax  
Write Stats-Record  
Close Statistics-File
```

- The write statement inserts the new record in the correct record number.
- You can use Invalid Key/Not Invalid Key to check if the write was successful.

```
Procedure Division.  
Open I-O Statistics-File  
Move 3 to Stats-Record-Number, Month-Number  
Move 12.50 to Monthly-Total-Gross  
Multiply Monthly-Total-Gross by 0.25  
Giving Monthly-Total-Tax  
Write Stats-Record  
Not Invalid Key Display "Record Inserted"  
End-Write
```

Syntax - Rewrite Statement:

The Rewrite command is used in COBOL to update an existing record in a relative file.

- The file can be opened in Random, Dynamic or Sequential access mode.
- The file must first be opened for I-O before using the rewrite statement.
- If the access mode is sequential, a record must first be read. This record is then updated by Rewrite.
- If the access mode is Dynamic or Random, a record can also be read, updated and re-written. This is a common way of updating records.
- However, for dynamic or random access, the relative key can be specified and new values moved into the record descriptor without a prior read.

```
Procedure Division.  
Open I-O Statistics-File  
Read Statistics-File  
Rewrite Stats-Record
```

```
Procedure Division.  
Open I-O Statistics-File  
Move 3 to Stats-Record-Number  
Read Statistics-File  
Move 11 to Month-Number  
Rewrite Stats-Record
```

```
Procedure Division.  
Open I-O Statistics-File  
Move 3 to Stats-Record-Number  
Move New-Record to Stats-Record  
Rewrite Stats-Record
```

Syntax - Sequential Delete:

Records can be deleted using the Delete command. How the Delete command works is different for sequential and random or dynamic access modes.

- Before deleting a record, a record must be read from the file.
- The last record that was read from the file will be deleted by the Delete statement.
- You can use File Status to determine whether the deletion was a success or failure.

```
Procedure Division.  
Open I-O Statistics-File  
Read Statistics-File  
Delete Statistics-File  
If (Status-Area = 0)  
    Display "Record Deleted"  
End-If
```

Syntax - Direct Delete:

The Delete statement is also used to delete records when accessing a relative file directly. In this case, the record number of the record to be deleted is specified. The format of the Delete statement is the same for random or dynamic access as it is for sequential access.

- Unlike sequential access, it is not necessary to read a record before deleting it.
- The record number of the record to be deleted is moved to the relative index specified in the File-Control definition.
- If the specified key's record is not found, no record is deleted.

```
Procedure Division.  
Open I-O Statistics-File  
Move 3 to Stats-Record-Number  
Delete Statistics-File  
    Not Invalid Key Perform Stats-Record-Success  
End-Delete
```