**VSAM Notes**

**sites**

**Tutorials**

**Example Programs**

-----------------------------------------------------------------------------------------------------------------------------

**Imp Points**

* **Common VSAM abends are 10, 22, 23, 35, 92, 94, 96, 97**

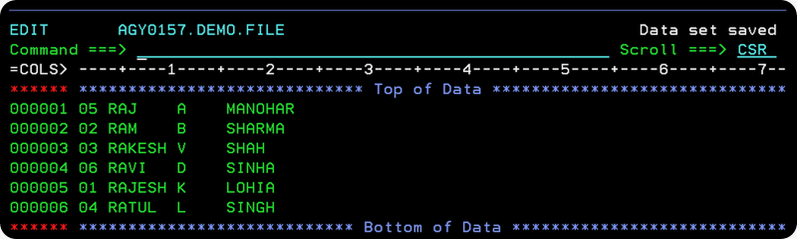
-----------------------------------------------------------------------------------------------------------------------------

**Questions and their Explanation**

**Q. What is VSAM? Never heard of it…!**

VSAM stands for Virtual Storage Access Method. In Windows, user's data is stored in files. In a text file, on Windows or Linux, the data consists of lines/records one after the other. Such files are called Sequential Datasets (Files). VSAM is a new, improved way of storing Data. VSAM overcomes some of the limitations of conventional file systems like Sequential Files.

**Q. Hold on for a sec... What does a Sequential File look like?**

In the early days of computers, all the data was stored in Sequential Files(Physical Sequential Dataset). Data was stored in the form of records, one after the other. Suppose, we wanted to store the information about all the employees in our organization. Below, you’d find a find picture of a how a Sequential File/PS Dataset looks like :     
      
[](http://lh5.ggpht.com/_sQvdFWqMlMg/S4p-ZBFu8hI/AAAAAAAACk4/r_dCljNe-BU/s1600-h/Image151%5b1%5d%5b4%5d.png)  
  
As you can see, each record represents the data of a single, individual employee. This way, there would be thousands of records that make **EMPDAT Sequential File.**

**Q. So, its pretty cool the way a Sequential File stores data. But how to get it back? How to search for a particular employee?**

Well, that’s the tough part. Coz sequential datasets work more or less similar to a Cassette Tape. Yup.. an audio cassette tape. The songs recorded on the cassette tape are analogous to records in a Sequential File.If you want to play a particular song, you have to start from the beginning of the tape, travel through the entire tape, till you reach the desired song. You can’t directly jump to a song and play it. You have to read through the tape, and forward scan through it, till you reach the desired place.   
  
On the same lines, when you want to search for a particular record say Employee no. 04, you have to travel through the entire the list of records, one by one, till you reach the desired record. The longer is the Sequential File, the longer is would take to access the record. You just don’t know, where the record lies hidden in such huge list or sequential file. The records are scattered and distributed hap-hazardly in the file. So, searching or getting data records, i.e.retrieval of data in a sequential file takes a very long time.

**Q. I get it.. as far as Searching  goes – Sequential Files are not very efficient. What’s VSAM got to offer?**

You can use a more structured and organised way of storing this data called **VSAM**. Though the abbreviation is a little geeky, VSAM files are superior in comparison to ordinary sequential files. Searching and retrieving data from a VSAM File is very fast.   
Apart from this, there are many other advantages that VSAM has to offer :   
- Free space in a VSAM File is not wasted, it is reclaimed automatically.   
- VSAM Files are device and O/S independent, this means if you stored data in VSAM on MVS O/S on Mainframes, you can port the file, and read it from Windows O/S on an Intel Machine, without impacting the data contents.

**Q. What are the types of VSAM files?**

VSAM files are of 3 types -   
1) Entry Sequenced file(ESDS)2) Key Sequenced files(KSDS)   
3) Relative Record file(RRDS)   
  
**A VSAM file is also called a** **Cluster**. Hence, the names ESDS Cluster, KSDS Cluster and RRDS cluster, are used interchangeably with ESDS File, KSDS File, RRDS File.

**Q. What’s a Key Sequenced File(KSDS)? Can you explain in brief?**

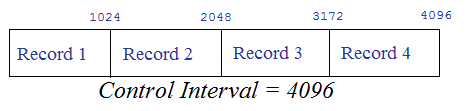
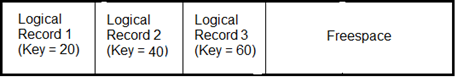
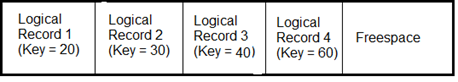
- **Concept of Key** : In a KSDS file, every record is identified by a unique identification key**.** Every single, individual employee will have a distinct and unique key value**.** This key could be his Employee Identification No, since it is unique for each employee. No two employees can have the same key value. **- How data is stored in a KSDS File :** When you first create a KSDS file, it is initially empty. You must fill data into the KSDS file. Thus, you need to populate(Load) the KSDS file with real data.Generally we do a sequential load, which means the data must be supplied inincreasing(ascending) order of the key. This is because, a KSDS file stores all the data records in increasing(ascending) order of the key.

**- For Dummies - Concept of Memory Address** :    
A KSDS file has 2 parts **–** Index ComponentandData Component**.** The Data Component contains the Data records. Every record is stored in 1 Storage or memory Location. Every memory location houses 1 record**.** Just like, the houses on a street in which people live, in Mainframe memory, in each house/cell/storage location lives 1 record.Houses on a street have a residential address by which they can be easily reached. If you know the house address, you can access the house. The same way, our houses/storage locations in the memory have unique addresses, by which they can be accessed. If you knew the location/address of a memory location you can easily access the record stored there(in much less time).

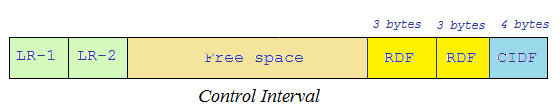
**- For Dummies – Comparing a Book’s Index with a KSDS Index ; How search performance improves with the help of Index Component** :    
Imagine, if you didn’t have an index in a book, and you wanted to find a keyword. You would have to read through the entire length of the book, page by page, till you come across the word you’ve been looking for. The Index simplifies this activity. Basically, a book index has two columns, one the keyword, and other the page no./location in the text where this keyword is located. Every page has a page-number. Let’ say you want to search the term Mainframe Computers. You look up this keyword in the Index. This is easy, because the index is sorted in Alphabetical order of the key-value. You jump to the section –'M'. Look up this term, in the index points to Page No. 373. You jump straight to page 373 and start reading about Mainframe Computers.   
  
Just as every page has a page no., every record in a KSDS Data file has an address. **The KSDS Index file has an entry for every key-value(key-field)**. For example, employees 1, 2 and 3 each would have an entry in the index. The index also stores the memory address(offset) of this Employee record in the KSDS Data file.   
  
Like a book index is sorted alphabetically on the keyword, the KSDS index file is sorted in increasing order of the key-field. Let's assume, Employee ID as key-field. So, how does it work? Let's say you wanted to find the name of Employee No. 0004. Simple, you look up the the row of Employee, with Key-value=0004 in the KSDS Index file. This is easy, because, the index is already sorted on the Key field => Employee ID. Now, you find the address of the Storage Location(House) in the KSDS Data file, where Employee ID 04 lives.This is location no. 600**.**Since you know the address, you can now directly jump and fly to address 600, and access the name of the Employee. This is far quicker than you thought.   
  
**The gist of this concept is, KSDS Index file stores key-values, and pointers(memory address)set to the corresponding records in the Data file. This way, Searching is faster and easier.** **The process of building an Index on a key-field for Data Records is called Indexing(or simply building an INDEX).**   
  
Let me caution you, that the diagram above is a very crude or preliminary picture of the KSDS Index file. Don’t go by it. **In reality, the KSDS Index file has an inverted-tree structure. In Computer Science, we call such a tree, a B+ Tree**. If you are curious to know, what’s a B+ Tree, and how the KSDS Index file really looks, read on. If you feel, you’ve absorbed a lot, you can call it a day!

**Q. How records are organized in KSDS Data file?**

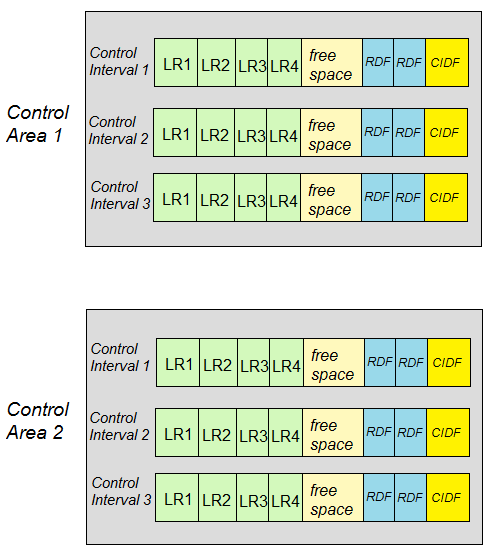
A KSDS file stores logical records of a file in fixed length blocks called Control Intervals(CI).In a KSDS Data file, a Control Interval holds several logical records. The logical records within each control-interval are always kept sorted by key-field.A KSDS File could have thousands of Control Intervals. In a Control Interval, records can be of any size or length. We do not distinguish in particular between fixed-length and variable-length records. However, as a rule, all Control Intervals in KSDS file are exactly equal in size(length).   
  
When a new KSDS file is created, you must specify the size of the Control Intervals in the file. By default, the Control Intervals(CI) in a KSDS File assume a size= 4k(4096) bytes. However, the size of Control Intervals in KSDS Files can lie in the range of 512 bytes <= Control Intervals Size <= 32k   
  
When you create a new KSDS File, the control intervals in it are empty. As you load data into the KSDS file, the Control Intervals are populated with information.   
  
What follows from hereon, shall give you a picture of how Control Intervals look like in Memory.

**Control Interval (Very idealistic – Simplified)**Assume that, Control Intervals are 4096 bytes long. A logical record(Employee record) spans 1024 bytes. Then,   
  
No. of records per CI = 4096/1024 = 4 records/CI   
  
Thus, in this example, the Control Interval is completely full(no room for new records).   
  
**    
  
**Control Interval often contains some empty/free space(Close to real model)** :   
Assume that, Control Intervals are 4096 bytes long. The first logical record = 1000 bytes, the second logical record = 1500 bytes, the third logical record= 1,300 bytes.   
  
Logical Record 1 + Logical Record 2 + Logical Record 3   
= 1000 + 500 + 1300 = 2800 bytes.   
  
Thus, the remaining space = 4096 – 2800 = 1,296 bytes is left free. This free-space can be used to accommodate a new logical record. Thus, Control Intervals may also have free-space.   
  
New logical records can be added to a Control Interval, by using the free-space in the Control Interval(CI).   
  
**[](http://lh5.ggpht.com/_sQvdFWqMlMg/S4qOSYxlDUI/AAAAAAAAClI/77or3-vdq9A/s1600-h/Image153%5b1%5d%5b4%5d.png)**  
VSAM Control Interval   
  
[](http://lh4.ggpht.com/_sQvdFWqMlMg/S4qOUlmDvFI/AAAAAAAAClQ/3x1PnQErFyA/s1600-h/Image152%5b1%5d%5b4%5d.png)  
Control Interval showing addition of record with key 30   
  
Let's look at the recipe followed by VSAM, to add a new logical record to a KSDS File.

1. VSAM goes through a full-index search to locate the Control Interval(CI) in the KSDS Data file, in which the new record must be placed.   
(This search is exactly the same as that used to randomly retrieve a record).   
2. After the index search locates the Control Interval(CI), that Control Interval(CI) is loaded into memory. VSAM then searches through the logical records in the Control Interval to determine, where the new record should go.(Recall, that a KSDS file stores all data records in increasing order of the key).   
3. The new record is then inserted into the Control Interval(CI), in key sequence, re-arranging the other records, as necessary.   
4. The updated Control Interval(CI) is now written back to its original location on the Disk.

**Control Interval also contains extra Information(Real Model)** :    
**VSAM treats all the logical records, as if they were variable-length(even if, they are fixed-length). VSAM keeps track of the length of Logical records in a Control Interval**, by using special **Record-definition Field(RDF)**, at the end of each Control Interval. This special field that holds the length information for each logical record is 3 bytes long.   
  
Moreover, VSAM also keeps track of the amount of the free-space and its location, within a Control Interval. This meta-information is stored in a special **Control Interval-definition Field(CIDF),** at the end of each Control Interval. This special field that holds [amount,location] of the free-space for a Control Interval is 4 bytes long.  **    
  
**Control Area(CA)** :   
A Control Area(CA) is a group of related Control Intervals.    
  
KSDS Files are organised as Control Areas(CA) which in turn contain hundred’s of fixed-length Control Intervals(CI) filled with logical records, free-space and Control information.

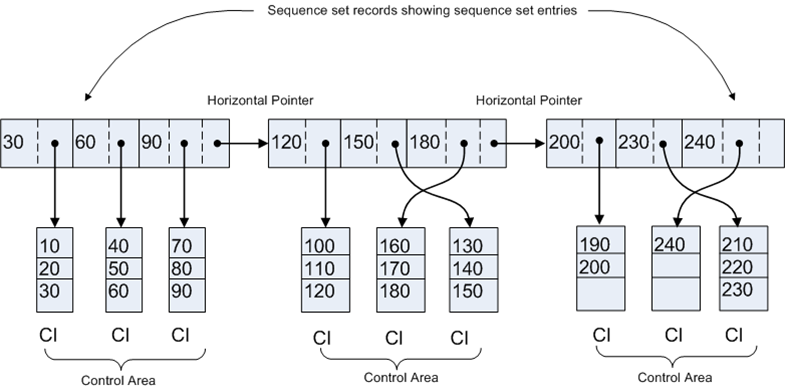
**Q. Can you show me a picture or visual of how KSDS Data file looks like?**

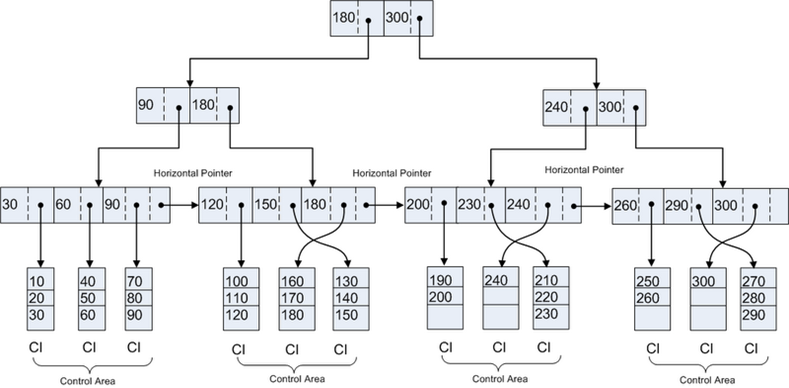
A KSDS Data file is – a collection of control intervals and control areas. A CI normally holds several logical records. At the end of each CI, control information is stored. Between the logical records, and the control information, there’s free-space, where new records can be added.   
  


**Q. What does a KSDS Index file look like?**

The KSDS Index file is organised in two parts – Index Set and Sequence Set. Lowest level of index entries is called the Sequence set. There is one sequence set record for each control area, in the KSDS Data file. The sequence set record for each control area, contains an entry for each control interval in that control area. The entry for a control interval stores (i) the highest key of the logical records in that CI (ii) the physical disk address(pointer to) of that CI.

The CI entries within a sequence set record, are kept in increasing(ascending) order of the key. This facilitates control-intervals within a control-area to be retrieved in key-sequence, during sequential processing, irrespective of whether the actual CI’s are in key sequence within the CA.

As I just said, the sequence set record for a CA, contains an entry for each control interval in that control area. In order to facilitate random processing, each CI entry has a (i) highest key of the CI (ii) vertical pointer to the Control Interval. The vertical pointer can be followed to retrieve any or all the records within that CI.   
  
In addition to vertical pointers to each CI, each sequence set record also contains a horizontal pointer, to the next sequence set record in key sequence. The horizontal pointers are followed during sequential processing. After all the records in a control area have been read, the horizontal pointer is followed to move to the next sequence set record which points to the successive control area.   
  


The Index set is organised as a tree or hierarchical structure. There is one and only one index set record at the top of the tree(that is at the root). Index searching during random processing begins at this root index set record.   
  
The root and all the other index set records consist of several entries. Each entry consists of the highest key of the next lower-level index set record, and a pointer to said index set record. The individual entries within an index set record are kept in key sequence.   
  
During Random processing, the logical record that you want to access, must be first looked up in the Index. This process proceeds as follows :   
1. The root index set record is input, and the first entry greater than or equal to the key of the desired record is located. Associated with this key value, is downward pointer to next lower-level index set record.   
2. The next lower level index set record is input, and the first entry >= key of desired record is located. Associated with this key value, is a downward pointer to the next lower-level index set record.   
3. This process continues, until you reach a sequence set record. At this point, the first CI entry >= key of the desired record is located. Associated with this key value, is a downward pointer to the control-interval.   
4. The indicated control-interval(CI) is input, and is searched for desired logical record. If the record is not in this CI, it is not in the file(and the COBOL program is notified of record-not-found condition).   
  
[](http://lh6.ggpht.com/_sQvdFWqMlMg/S5By1xsy71I/AAAAAAAACmo/rXupmzc_VRk/s1600-h/Image155%5b1%5d%5b5%5d.png)

**VSAM INTERVIEW QUESTIONS AND ANSWERS**

**1.What are the different types of VSAM files available?**

ESDS: **Entry Sequence Data Set**

KSDS: **Key Sequence Data Set**

RRDS: **Relative Data Set**

**2. What is IDCAMS ?**

IDCAMS is the **Access Method Services** program. You run the IDCAMS program and supply AMS commands thru SYSIN. (examples of AMS commands are DELETE, DEFINE, REPRO etc..).

3. Can AMS commands be run from the TSO prompt ?

Yes

**4. Syntax of AMS modal commands ?**

Note: these can be used only under IDCAMS and not from the TSO prompt.

IF LASTCC(or MAXCC) >(or <,= etc..) value -

THEN -

DO -

command set (such as DELETE, DEFINE etc..)

ELSE -

DO -

command set

LASTCC - Condition code from the last function(such as delete) executed

MAXCC - Max condition code that was returned by any of the prev functions

SET is also a valid AMS command. SET LASTCC (or MAXCC) = value

The maximum condition code is 16. A cond code of 4 indicates a warning. A cond code of 8 is usually encountered on a DELETE of a dataset that is not present.

**5. Under IDCAMS , multiple functions can be executed, each of which returns a cond code. What will be the condition code returned to the operating system ?**

The maximum condition code generated is returned as the condition code of the IDCAMS step.

**6. What is Control Interval, Control Area ?**

Control Interval is analogous to a physical block for QSAM files. It is the unit of i/o. Must be between 512 bytes to 32 k. Usually either 2K or 4K. **A larger control interval increases performance for sequential processing while the reverse is true for random access**. **Under CICS when a record is locked, the entire CI gets locked.**

Control area is a group of control intervals. CA is used during allocation. CA size is calculated based on the allocation type (cyl, tracks or records) and can be max of 1 cylinder

**7. What is FREESPACE ?**

Coded in the DEFINE as FREESPACE(ci ca) where ci is the percentage of each control interval to be left free for insertions, ca is the percentage of control intervals in each control area to be left empty.

**8. How do you decide on optimum values for CI, FREESPACE etc..?**

**CI size should be based on record length, type of processing**. Usually CI is 4K. If record length is larger(>1K), chose 6K or 8K.

FREESPACE should be large if more number of insertions are envisaged. Usual values are (20 20) when heavy updates are expected. CI size can be calculated.

**9. Would you specify FREESPACE for an ESDS?**

No. Because you cannot insert records in an ESDS, also when you rewrite a record, it must be of the same length. Thus putting any value for freespace does not make any sense.

**10. What is SHAREOPTS ?**

SHAREOPTS is a parameter in the DEFINE and specifies how an object can be shared among users. It is coded as SHAREOPTS(a b), where a is the cross region share option ie how two or more jobs on a single system can share the file, while b is the cross system share option ie how two or more jobs on different MVSes can share the file. Usual value is (2 3).

**11. What is the meaning of each of the values in SHAREOPTS(2 3)?**

Value of 2 for cross region means that the file can be processed simultaneously by multiple users provided only one of them is an updater. Value of 3 for cross system means that any number of jobs can process the file for input or output (VSAM does nothing to ensure integrity).

**12. How do you define a KSDS ?**

DEFINE CLUSTER(cluster name) with the INDEXED parameter. Also specify the ds name for the DATA component & the ds INDEX component. Other important parms are RECORDSIZE, KEYS, SHAREOPTIONS.

**13. How do you define an ALTINDX ? How do you use ALTINDXs in batch, CICS pgms ?**

DEFINE ALTERNATEINDEX. Important paramters are RELATE where you specify the base cluster name, KEYS, RECORDSIZE,SHAREOPTIONS,UNIQUEKEY(or NONUNIQUEKEY), DATA(ds name for the data component), INDEX(ds name for the index component).

Then DEFINE PATH. Important paramters are NAME (ds name for the path), PATHENTRY (ds name of the alternate index name), UPDATE(or NOUPDATE) which specifies whether an alt index is updated when a update to the base cluster takes place.

Then BLDINDEX. Parameters are INDATASET(ds name of base cluster), OUTDATASET(ds name of AIX).

Using alternate indexes in batch pgms:

In the JCL, you must have DD stmts for the cluster and for the path(s). In the cobol pgm, SELECT .. ASSIGN TO ddname for base cluster RECORD KEY IS... ALTERNATE RECORD KEY IS..

Using alternate indexes in CICS pgms:

FCT entries must be created for both base cluster & the path. To read using the alternate index, use the dd name of the path in CICS file control commands.

**14. What happens when you open an empty VSAM file in a COBOL program for input?**

**A VSAM file that has never contained a record is treated as unavailable**. Attempting to open for input will fail. An empty file can be opened for output only. When you open for output, COBOL will write a dummy record to the file & then delete it out.

**15. How do you initialize a VSAM file before any operation? a VSAM with alternate index?**

Can write a dummy program that just opens the file for output & then closes it.

**16. What does a file status of 02 on a VSAM indicate?**

**Duplicate alternate key** . Happens on both input and output operation

**17. How do you calculate record size of an alternate cluster? Give your values for both unique and non-unique.**

**Unique Case**: 5 + ( alt-key-length + primary-key )

**Nonunique Case**: 5 + ( alt-key-length + n \* primary-key )

where n = # of duplicate records for the alternate key

????Any one who knows - can you explain ?

**18. What is the difference between sequential files and ESDS files?**

**Sequential(QSAM) files can be created on tape while ESDS files cannot.**

**Also, you can have ALTINDEX for an ESDS while no such facility exists for QSAM files.**

**19. How do you load a VSAM data set with records ?**

Using the REPRO command.

**20. How do you define a GDG ?**

Use the DEFINE GENERATIONDATAGROUP command. In the same IDCAMS step, another dataset must be defined whose DCB parameters are used when new generations of the GDG are created. This dataset is known as the model dataset. The ds name of this model dataset must be the same as that of the GDG, so use a disp of keep rather than catlg and also specify space=(trk,0)

**21. Do all versions of the GDG have to be of the same record length ?**

**No, the DCB of the model dataset can be overridden when you allocate new versions.**

**22. How are different versions of GDG named ?**

base-file-name.GnnnnnV00 where nnnn= generation number (upto 255).

nnnn will be 0000 for the 1st generation.

**23. Suppose 3 generations of a GDG exist. How would you reference the 1 st generation in the JCL?**

Use GDG name(-2).

**24. Suppose a generation of GDG gets created in a particular step of a proc. How would you refer the current generation in a subsequent step? What would be the disposition of this generation now?**

Relative generation numbers are updated only at the end of the job, not at the end of a step. To allocate a new generation, we would be using (+1) with a DISP of (NEW,CATLG,DELETE). To refer to this in a subsequent step in the same job, we would again use (+1) but with a DISP of SHR or OLD.

**25. What more info you should give in the DD statement while defining the next generation of a GDG?**

Give (+1) as the generation number, give (new,catlg) for disp, give space parameter, can give the dcb parameter if you want to override the dcb of the model dataset.

**26. Assuming that the DEFINE jcl is not available, how do you get info about a VSAM fileï¿½s organisation ?**

Use the **LISTCAT** command

**27. During processing of a VSAM file, some system error occurs and it is subsequently unusable . What do you do ?**

**Run VERIFY.**

**ABENDS**

**00**- SUCCESSFUL COMPLETION

**02**- DUPLICATE KEY, NON UNIQUE ALT INDEX

**04**- READ, WRONG LENGTH RECORD

**05**- OPEN, FILE NOT PRESENT

**10**- END OF FILE

**20**- INVALID KEY VSAM KSDS OR RRDS

**21**- SEQUENCE ERROR, ON WRITE OR CHANGING KEY ON REWRITE

**22**- DUPLICATE KEY

**23** - RECORD NOT FOUND - (when we are trying to access a record with key)

or

FILE NOT FOUND

**35** - OPEN, FILE NOT PRESENT

When we will use this code in our program?

There are situations where file should be read if exists, write if it does not

when you dont know whether file exists are not , first you will open

file in I-O mode and check status code. if it is 35 then open that

file for output file. other wise you will continue with your logic

**41** - OPEN, FILE IS OPEN

**42** - CLOSE, FILE IS CLOSED

**43** - DELETE OR REWRITE & NO GOOD READ FIRST

**46** - SEQUENTIAL READ WITHOUT POSITIONING

**47** - READING FILE NOT OPEN AS INPUT/IO/EXTEND

**48** - WRITE WITHOUT OPEN IN IO MODE

**49** - DELETE OR REWRITE WITHOUT OPEN IN IO MODE

**92** - LOGIC ERROR/OPENING AN OPEN FILE

OR READING OUTPUT FILE

OR WRITE INPUT FILE

OR DEL/REW BUT NO PRIOR READ

**94** - SEQUENTIAL READ AFTER END OF FILE

OR NO CURRENT REC POINTER FOR SEQ

**96** - MISSING DD STATEMENT IN JCL

**97** - OPEN OK, FILE INTEGRITY VERIFIED

When we will use this in our programs?

We use this code whenever we open the file, if status code is 00 or 97

we will proceed with our logic, other wise, call error routine.

Usaully, it may come when file was not closed.

for example

IF WS-FILE-STATUS NOT = '00' AND '97'

PERFORM ERROR-ROUTINE

END-IF.