VSAM (**V**irtual **S**torage **A**ccess **M**ethod)

History

Access Method is an interface between the application program and physical operation of storage devices. It is a component of operating system. VSAM is the first access method that efficiently uses the virtual storage of MVS. It can manipulate only the data that resides on a DASD. (Direct access storage device)

IBM introduced VSAM in 1973 as a replacement for several existing access methods designed by it earlier.

KSDS (Key Sequenced Data Set) replaced ISAM (Indexed Sequential Access Method) RRDS (Relative Record Data Set) replaced BDAM (Basic Direct Access Method)
ESDS (Entry Sequence Data Set) provide same function as normal sequential QSAM. (Queued Sequential Access Method).

Initially VSAM had only ESDS and KSDS. RRDS and Alternate Index to KSDS are introduced in 1979. DF/ EF (Data Facility extended Function) VSAM was introduced in 1979 with Integrated Catalog Facility (ICF) to replace the old VSAM catalog of the previous versions.

The latest version of DFP/ VSAM released in 1991 called DFP/VSAM 3.3 contains enhancements like variable record length support for RRDS and added DFSMS facilities.

Advantages of VSAM over other access methods

1. Data retrieval will be faster because of an efficiently organized index. The index is small because it uses a key compression algorithm.
2. Insertion of records is easy due to embedded free space in the cluster.
3. Records can be physically deleted and the spaces used by them can be used for storing other records without reorganization.
4. VSAM datasets can be shared across the regions and systems.
5. Datasets can be physically distributed over various volumes based on key ranges.
6. VSAM is independent of storage device types.
7. Information about VSAM datasets is centrally stored in VSAM catalog.
So referencing of any VSAM dataset need not be detailed in JCL.

Disadvantages of VSAM

6. To allow easy manipulation of records, free space should be left in the dataset and this increases the spaces required.
7. Integrity of the dataset across the region and system need to be controlled by user.

<u>CLUSTER</u>

A cluster can be thought of as a logical dataset consisting of two separate physical datasets:
1 The data component (contains the actual data).
2 The index component (contains the actual index).

All types of VSAM datasets are called clusters even though KSDS is the only type that fulfills the cluster concept. ESDS and RRDS don't have Index component.

<u>Data Component</u>.

<u>Control Intervals and Control Areas</u>

VSAM stores records in the data component of the Cluster in units called control intervals.

The control interval is the VSAM equivalent for a block and it is the unit of data that is actually transmitted when records are read or written. Thus many logical records form a control interval and many control intervals form a control area.

The Control Area (CA) is a fixed length unit of contiguous DASD storage
containing a number of CI s. The control area is VSAM internal unit for allocating space within a cluster. The primary and secondary allocations consist of some number of CA. Minimum size of control area is 1 track and maximum size is 1 cylinder.

<u>Format of Control Interval</u>

There are four separate areas within a control Interval.
1. Logical Record Area (LRA) that contains the records.
2. Free Space (FSPC). This area can be used for adding new records.
3.Unused Space. As FSPC may not be a multiple of record length, there will be always some unused space in every CI. This can be minimized for fixed length records by selecting proper control interval size.
4. Control Fields.

- ❖ CIDF – Control Interval Definition Field - 4 bytes field containing information on free space availability in the control interval. One per control interval.

- ❖ RDF – Record Definition Field – 3 bytes field. For fixed length records, there will be 2 RDF, first contains the number of records in the control interval and the second contains record length. For variable length records, the number of RDF can vary depending on how many adjacent records have the same length in the CI. If no two adjacent records are of the same length, then one RDF is needed to describe each record.

<u>Index Component (Sequence Set and Index Set)</u>

Besides the data component, VSAM creates an index component. Index component consists of index set and sequence set. The sequence set is the lowest level of the index is called the sequence set and it contains primary keys and pointers to the control intervals of the data component.

There is one sequence set for one control area. The highest record key of every control interval is stored in sequence set. The highest record key of the sequence set is stored in first level of index set. Based on the size of control interval of index component, there will be 1-3 levels of index sets in the index component of the dataset.

Control Interval and Control Area Split

When a VSAM dataset is created, the free space parameter defined can specify a percentage of control intervals to be left free during the load of VSAM file. Later, when you add a record to the VSAM file, according to the key sequence, it is placed in a specific control interval.

But if the specific control interval is already full, then this cannot be placed in the sequence. The result is Control Interval split. It moves half of the records in the filled control interval to any other free control interval available in the control area and makes room for the new record in the right place.

If there is no free control interval exist in the control area, then half the control intervals are moved to new control area and this is called control area split.

Example

1. In the example below, there are four control areas and every control area contains two control intervals. Control fields are not shown in the diagram. There should be one sequence for every control area. So there are four sequence sets. There are two levels of index set. The second level of index set contains pointers to sequence set.

2. Control Interval Split: When a record with key 22 is added in the program, it should physically be stored between the existing records 21 and 23. 21 and 23 are in the first control interval of control area-3. There is no more space available to store this record. So control split will occurs. Record 20 and 21 continue to exist in the current control interval. Records 22 and 23 will be moved to any of the free control intervals. In our case control interval 2 is free. So they are moved there and index set is accordingly updated.

3. Control Area Split: When a record with key 3 in the program, it should be placed between 2 and 4. These records are in first control interval of first control area and there is no free space. So control interval split is expected. But there is no free control interval in the control area-1. So control area split occur. New control area is allocated and half the records of control area-1 will be moved there and indexes are properly updated.

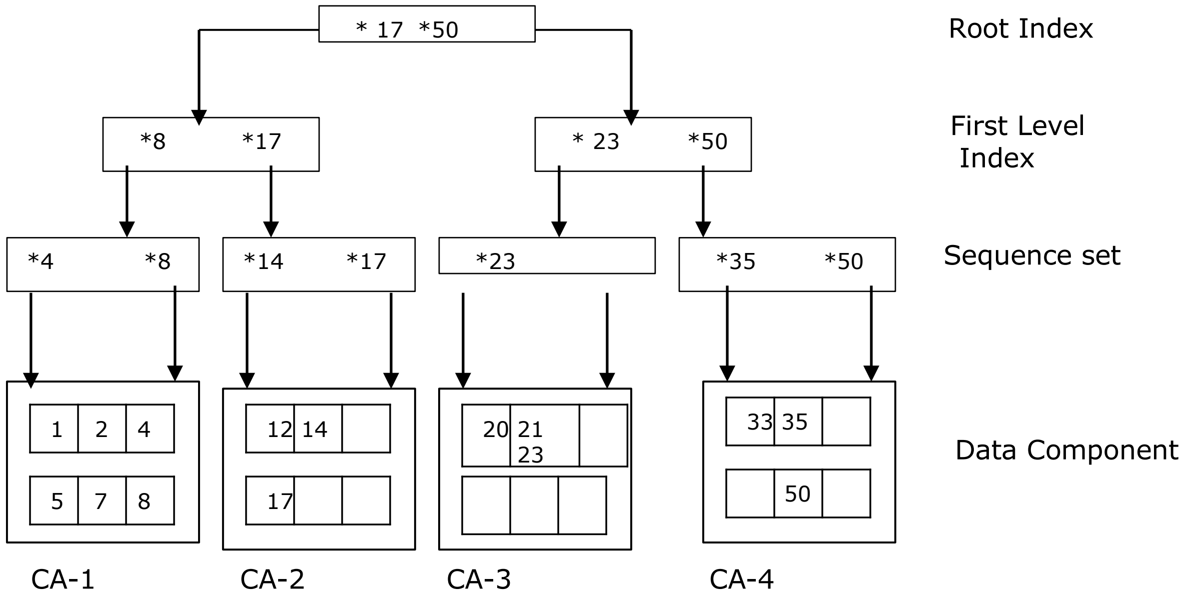
How index could make the access faster?

I faced an interesting question in an interview. The question follows:

My sequential file has 50 records. To get the 50th record, I have to read 49 data records and bypass. Indexes store primary keys with actual location. So to read 50th record using indexed organization, I have to get the location of 50th record from index. So I have to read and bypass the location of 49 records. The only difference is in the second case is I am doing sequential read in index set instead of dataset. How do you say my index access would be faster?

To answer this question, you should know how indexes are organized and read. We have already seen how they are organized. In the example below, to read the 50th record, first root index is read and identified that the record should be in the right hand side. In the second I-O, I will get the sequence and in third I-O, I will get the location of the record. I will get my record in the fourth I-O instead 51 I-O. (50 IO in index and 1 I-O for getting the data)

If I am accessing the first record, then sequential read needs only one I-O but obviously my random read needs more. So we prefer indexed organization only when the number of records is significant.



<u>Properties of VSAM datasets</u>

Properties of ESDS, KSDS and RRDS are listed a tabular format in the next page. Though the table says RRDS records should be of fixed length, MVS/DFP VSAM Version 3.3 allows variable length RRDS. VSAM internally implements it using KSDS.

VSAM LDS Properties: Linear datasets have no records. They are just long strings of bytes. Since it is a long string of bytes an LDS has no FSPC, US, CIDF and RDF. The CISZ is always 4k bytes. The main use of LDS is to implement other data access organizations especially a relational database system like DB2.

As the data in an LDS is manipulated the operating system automatically pages in and out the portions of the dataset being worked on. The dataset is addressed by the RBA as if it were in memory and the system pages the needed pages in and out. Thus the process is very simple and fast.

ESDS differs from QSAM file in the following ways:

1. QSAM can be created over TAPE whereas ESDS cannot reside on TAPE.
2. Alternate Index can be created over ESDS. This alternate index has alternate key mapped with RBA. So Indexed and Random access are possible using the alternate Index in ESDS whereas it is not possible with QSAM.

Comparison of ESDS, KSDS and RRDS

Characteristics	ESDS	KSDS	RRDS
Entry Sequence	Based on entry sequence	Based on collating sequence by key field	Based on relative record number order
Access	Only Sequential access is possible.	Sequential and random access is possible. Random access is thru primary/alternate key.	Can be accessed directly using relative record number, which serves as address.
Alternate INDEX	May have one or more alternate indexes. But cannot be used in BATCH COBOL. Can be used in CICS COBOL.	May have one or more alternate indexes.	Not applicable
Location of the record	A record RBA cannot be changed	A record RBA can be changed.	A relative record number can be changed.
Free Space	Exist at the end of the dataset to add records.	Distributed free space for inserting records in between or change the length of existing record.	Free slots are available for adding records at their location.
Deletion	Cannot be deleted. REWRITE of same length is possible.	DELETE is possible.	DELETE is possible.
Record Size	Fixed or Variable	Fixed or Variable	Fixed.
SPANNED records	Possible	Possible	Not possible
Speciality	Occupies less Space	Easy RANDOM access and most popular method	Fastest access method.
Preference	Application that require sequential access only. Ex: PAYROLL PROCESSING.	Applications that require each record to have a key field and require both direct and sequential access. Ex: Banking application	Applications that require only direct access. There should be a field in the record that can be easily mapped to RRN.

IDC Access **M**ethod **S**ervices (IDCAMS)

AMS is used to perform various functions on VSAM datasets and catalogs. AMS has got a utility program called IDCAMS. The functions of AMS are performed using the different functional commands of IDCAMS. It can be invoked in the following three ways:

1. Batch mode with JCL statements.
2. Interactively with TSO commands.
3. Calls from an application program.

Important functional commands of IDCAMS are:

1. DEFINE – To create objects – KSDS/ESDS/RRDS/GDG/VSAMSPACE etc
2. ALTER - To alter the parameters of the object already exists.
3. PRINT - To print and view the selected records of dataset.
4. DELETE – Delete the objects.
5. LISTCAT – View the complete information about any object.
6. REPRO - Copy/Restore/Merge Utility for VSAM and NON-VSAM files.
7. EXPORT and IMPORT – Copy and restore datasets across the system.
8. VERIFY – To properly close the unclosed/ABENDED VSAM dataset.

Sample IDCAMS JCL

```
   //JS10        EXEC PGM=IDCAMS,REGION=1024K, PARM=parameters
   //STEPCAT       DD DSN=..,DISP=SHR Optional STEPCAT
   //SYSPRINT  DD SYSOUT=*             IDCAMS Messages
   //SYSIN DD *
     Control statements
 /*
 //
```

Guidelines for coding commands

1.At least one blank space must be there between the command and the object and between sub parameter values.
2. A space is optional between a parameter and the parenthesis enclosing its values.
3. Sub-parameter values can be separated using a space or a comma.
4.Multiple parameters can be coded on a line but it is better to place each on a line by itself for better readability.
5.A parameter and a sub-parameter cannot be separated with a hyphen. A plus sign should be used for this purpose.

Meaning of FILE and DATASET in AMS commands

6.Keyword FILE should point to DDNAME and it is logical. There should be mapping of the DDNAME with actual dataset. JES allocates the datasets associated with the file just before the execution of step.

2.Keyword 'DATASET' should directly point to the physical dataset and IDCAMS allocates this file dynamically for operation.

DEFINE CLUSTER
This command is used to create and name a VSAM Cluster.

Basic Parameters for Define Cluster

DEFINE CLUSTER-NAME
This parameter specifies name to the VSAM cluster. The cluster name becomes the dataset name in any JCL that invokes the cluster

```
// INPUT DD DSN= SMSXL86.TEST.VSAM, DISP=SHR
```

The name for a VSAM dataset can include up to 44 alphanumeric characters.

When the data and index parameters are coded to create the data and index components, the name parameter is coded for them as well. If the name parameter is omitted for the data and index VSAM tries to append part of .DATA or .INDEX as appropriate as the low level qualifier depending on how many characters the dataset name contains already and still staying within the 44 character limit.
If data and index components are named, parameter values can be applied separately. This gives performance advantages for large datasets.

DEFINE CLUSTER-DATA COMPONENT
The data parameter instructs IDCAMS that a separate data component is to be created. Data is optional but if coded must follow all parameters that apply to the cluster as a whole. There are several options for the data parameter but name is the most common.

DEFINE CLUSTER-INDEX COMPONENT
The index parameter creates a separate index component. Index is optional
but if coded must follow all of the parameters that apply only to the data component. ESDS and RRDS must not have INDEX part.

DEFINE CLUSTER-SPACE Allocation
A VSAM dataset can have 123 extents in a VOLUME. Primary space is allocated initially when the dataset is created and based on request secondary space will be allocated. In the best case, you will get (primary +122 * secondary).
Refer the JCL section-SPACE parameter section for more details on EXTENTS.

VSAM calculates the control area size internally. Control area can of one cylinder, the largest permitted by VSAM, usually yields the best performance. So it is always better to allocate space in cylinders because this ensures a CA size of one cylinder.

The RECORDS parameter is used to allocate space in units of records for small datasets. When this is done the RECORDSIZE parameter must be specified.
If allocation is specified in units of KILOBYTES or MEGABYTES VSAM reserves space on the minimum number of tracks it needs to satisfy the request.

Syntax: UNIT(primary secondary)
UNIT can be CYL/CYLINDERS TRK/TRACKS REC/RECORDSD
KB/KILOBYTES MB/MEGABYTES

DEFINE CLUSTER-VOLUMES Parameter

The volumes parameter assigns one or more storage volumes to the dataset. Multiple volumes can be specified and if so, they must be of the same device type. The data and index components can be stored in separate volumes to provide a performance advantage for large datasets.

VOLUMES(volser) OR VOLUMES(volser1 volser2)

DEFINE CLUSTER-RECORDSIZE Parameter (RECSZ)

The record size parameters specify VSAM what records to expect. The AVG and MAX are the average and maximum values for variable length records. If records are of fixed length AVG and MAX can be the same. Record size can be assigned at the cluster or data level.

Syntax: RECORDSIZE(AVG MAX)

DEFINE CLUSTER- KEYS Parameter

The keys parameter defines the length and offset of the primary key in a KSDS record. The offset is the primary key's displacement in bytes from the beginning of the record. This parameter is defined for a KSDS only. Default is KEYS(64 0).

Syntax: KEYS(length offset)

DEFINE CLUSTER- Dataset Type Parameter

The dataset type parameter specifies whether the dataset is INDEXED(KSDS)
, NONINDEXED(ESDS), NUMBERED(RRDS) OR LINEAR (LDS).

INDEXED (IXD) specifies a KSDS and it is the DEFAULT. When this parameter is specified VSAM automatically creates and catalogs an index (if other parameters like keys are valid). Indexed is also used for a variable length RRDS.

NONINDEXED(NIXD) when specified denotes an ESDS. No index is created and records are accessed sequentially or by their relative byte address (RBA).

NUMBERED (NUMD) specifies an RRDS and LINEAR specifies an LDS.

Performance Parameters For DEFINE CLUSTER

The performance parameters are coded in the DEFINE CLUSTER command to

1. Reduce the amount of physical I-O.
2. Make necessary I-O faster.
3. Optimize disk utilization.

DEFINE CLUSTER- CONTROL INTERVAL SIZE Parameter

The size of the CI is specified by the CISZ parameter. The size of the CI is specified in bytes and it should be a multiple of 512 or 2048 depending on the type of catalog (ICF or VSAM) and the length of the record.

For datasets cataloged in ICF catalogs the control interval should be a multiple of 512 bytes with a range of 512 to 32768 bytes.

For datasets cataloged in VSAM catalogs, the data CISZ must be a multiple of 512 if records are of 8192 bytes or less, and a multiple of 2048 if records are of more than 8192 bytes. If a CISZ, which is not a multiple of the above two is assigned VSAM rounds the CISZ value up to the next highest multiple if necessary.

Best control interval size selection

For sequential processing of a KSDS, a relatively large CISZ will reduce physical I/O by keeping more records accessible in the buffers. On the other hand, for random processing of a KSDS a smaller CISZ would require lesser data transfer time and fewer buffers, thus making I/O faster.

For an ESDS (since it is processed sequentially) the CISZ should be relatively large depending on the size of the record.

Since an RRDS is processed randomly the CISZ should be relatively small.

DEFINE CLUSTER- SPEED And RECOVERY Parameters

RECOVERY pre formats a VSAM dataset at the time of initial load or resume load. It takes longer time to load the dataset through RECOVERY than through SPEED which does not pre format the dataset. However if the load operation is aborted in the middle, RECOVERY enables resumption immediately after the last record loaded.

If SPEED is specified load will have to be restarted from the beginning. SPEED is highly recommended because it will speed up the initial data load.

<u>The BUFFERSPACE Parameter (BUFSP)</u>

By default, VSAM allocates two data buffers for all types of datasets. One data buffer for processing and reserves the second for potential split activity. In addition to this, it would allocate one index buffer for a KSDS.

This parameter is used to override the default values. The BUFFERSPACE parameter represents the amount of storage in bytes required to process the contents of a minimum of one CI worth of data. Syntax: BUFFERSPACE(bytes)

For sequential processing more number of data buffers need to be allocated.

For random processing more index buffers may be required, at least one for each level of the index. For dynamic processing additional data and index buffers are required in proportion to the ratio of sequential and random processing planned.

The BUFSP value provided gets translated into data buffers for an ESDS or RRDS. But for a KSDS, VSAM decides on the number of data and index buffers based on the access method specified in the application program.

If BUFSP is specified in the DEFINE CLUSTER command, all applications that use the dataset can use only this allocation unless they override it in the BUFSP sub parameter of the Access Method Parameter of JCL.

<u>DEFINE CLUSTER-SPANNED Parameter (SPND)</u>

The SPANNED parameter allows large records to span more than one CI. If maximum record length is more than the CI size specified, allocation will fail unless the SPANNED parameter is specified. However records cannot span across control areas. The resulting free space in the spanned CI is unusable by other records even if they fit logically in the unused bytes. NONSPANNED is the default. It can be specified for ESDS and KSDS only.

<u>DEFINE CLUSTER-The KEYRANGES Parameter (KYRNG)</u>

The KEYRANGES parameter divides a large dataset among several volumes. The volumes in which the records are to be placed are specified using the VOLUMES parameter. Unless the KEYRANGES parameter is specified with the ORDERED parameter, the records are assigned randomly to the volumes. These parameters are illustrated by the following example

```
DEFINE CLUSTER                                                    -
(NAME(NTC1.V.US4.W2K00K.T30.AV.DW200006)                          -
CYL(5 1)                                                          -
KEYS(8 0)                                                         -
RECSZ(80 80)                                               -
KEYRANGES ((00000001 2999999)                              -
                (30000000 4700000)                                -
                (47000001 9999999))                               -
VOLUMES         (NTTSOB                                           -
                NTTSOJ                                            -
                NTTSO5)                                           -
ORDERED                                                           -
NOREUSE                                                           -
INDEXED
_ _ _ more parameters
```

When the ORDERED parameter is coded the number of VOLUMES and KEYRANGES must be the same.

DEFINE CLUSTER- REUSE Parameter (RUS)

This specifies that a cluster can be opened again and again as a reusable cluster. It means, whenever you open the dataset in OUTPUT mode, all the records that are already exist in the dataset are logically deleted. NOREUSE (UNRUS) is the default and specifies the cluster as non-reusable.

A cluster cannot be reused if
1. KEYRANGES parameter is coded for it
2. An alternate index is built for it
3. The cluster has its own data space in both the VSAM and ICF catalog environments.

DEFINE CLUSTER-IMBED And REPLICATE Parameters (IMBD/ REPL)

These parameters are applicable to a KSDS only. The IMBED parameter implies that the sequence set (lowermost level) of the index component of a KSDS will be placed on the first track of a data component CA and will be duplicated as many times as possible on that track.

What IMBED does for a sequence set REPLICATE does for an index set. The REPLICATE parameter forces each CI of the index set of the index component to be written on a separate track and replicated as many times as possible. This parameter reduces rotational delay when VSAM is accessing high-level index CI.

The IMBED option reduces the seek time it takes for the read-write head to move from the index to the data component and the replication factor reduces the rotational delay of the revolving disk.

NOIMBED(NIMBD) and NOREPLICATE(NREPL) are the defaults.

DEFINE CLUSTER-WRITECHECK Parameter (WCK)

This parameter instructs VSAM to invoke a DASD verify facility whenever a record is written. NOWRITECHECK (NWCK) is the default and provides no DASD verification. Since contemporary DASD devices are very reliable and because of the high performance overhead associated with this parameter it is better to accept the NOWRITECHECK option.

DEFINE CLUSTER-SHAREOPTIONS Parameter

This parameter specifies how a VSAM dataset can be shared across the regions and across the system. Default value is (1 3)

Syntax: SHAREOPTIONS(cross-region cross-system)

Type	Value	Meaning
CR	1	READ and WRITE Integrity. Any number of jobs can read the dataset OR only one job can write to the dataset.
CR	2	ONLY WRITE Integrity. Any number of jobs can read the dataset AND one job can write to the dataset.
CR CS	3	NO Integrity. File is fully shared. It is programmer responsibility to take proper lock over the file before use. Default value for CS.
CS	4	Same as 3 but additionally forces a buffer refresh for each random access.

Password Protection

VSAM datasets can be password protected at four different levels. Each level gives a different access capability to the dataset. The levels are

1. READPW- provides read only capability.
2. UPDATEPW- records may be read, updated , added or deleted at this level.
3. CONTROLPW- provides the programmer with the access capabilities of READPW and UPDATEPW.
4. MASTERPW- all the above operations plus the authority to delete the dataset is provided

Passwords provided at the cluster level protect only if access requires using the cluster's name as dataset name. Therefore it is advisable to protect the data and index components using passwords because someone could otherwise access them by name. Another feature of MVS called Resource Access Control Facility (RACF) ignores VSAM passwords and imposes its own security and for most VSAM datasets RACF security is sufficient.

The ATTEMPTS parameter coded with the password parameters specifies the number of attempts permitted for the operator to enter the password before abending the step.

The CODE parameter allows for the specification of a code to display to the operator in place of the entry name prompt.

The AUTHORIZATION parameter provides for additional security by naming an assembler User Security Verification Routine (USVR). The sub parameter for this enclosed in parenthesis is the entry point of the routine.

TO And FOR Parameters

When a dataset is allocated by Access Method Services the TO and FOR parameters are two mutually exclusive parameters given to specify the retention period of the cluster being defined.

TO(YYDDD) or
FOR(nnnn) YYDDD is Julian date & nnnn can be 1-9999

About CATALOG parameter

Most of the AMS commands provide CATALOG option. If the command is DEFINE, then the dataset being defined will be placed in the catalog, mentioned in the CATALOG parameter. Similarly while accessing the dataset,

1. The dataset is first searched in the catalog mentioned in CATALOG parameter.
2. If CATALOG is not coded, then the dataset is searched into the catalog coded in STEPCAT or JOBCAT catalog.
3. If there is no STEPCAT and JOBCAT, then the dataset is searched in the user catalog corresponding to the high level qualifier of the dataset.
4. If there is no user catalog found, then the dataset is looked into system catalog.
5. If the dataset is not listed in system catalog also, then you will get the error message of dataset not found.

There will be one master catalog and n number of user catalogs in the system. Every user catalog should have an entry in master catalog.

KSDS-ESDS-RRDS-LDS Sample Definitions

KSDS Definition

```
//KSDSMAKE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD*
 DEFINE CLUSTER                              -
 (NAME(EMPLOYEE.KSDS.CLUSTER) -
 VOLUMES(VSAM02)

 -
 CYLINDERS(2,1)

 -
 CONTROL INTERVAL SIZE(4096)

 -
 FREESPACE(10,20)

 -
 KEYS(9,0)

 -
 RECORDSIZE(50,50))

 -
 DATA
 -
 (NAME(EMPLOYEE.KSDS.DATA))
 -
 INDEX
 -
 (NAME(EMPLOYEE.KSDS.INDEX))    -
 CONTROL   INTERVAL   SIZE(1024)   -
 CATALOG(VSAM.USERCAT.TWO)
/*
```

ESDS Definition

```
//ESDSMAKE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD*
 DEFINE CLUSTER                              -
 (NAME (EMPLOYEE.ESDS.CLUSTER) -
  VOLUMES (VSAM02)             -
 CYLINDERS (2,1)

 -
CONTROLINTERVALSIZE (4096)

 -
RECORDSIZE (50,50)

 -
NONINDEXED)

 -
DATA

 -
(NAME (EMPLOYEE.KSDS.DATA))

 - CATALOG (VSAM.USERCAT.TWO)
/*
```

RRDS Definition

```
//RRDSMAKE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD*
 DEFINE CLUSTER                              -
 (NAME(EMPLOYEE.RRDS.CLUSTER) -
VOLUMES(VSAM02)              -
 CYLINDERS(2,1)
-
CONTROL INTERVAL SIZE(4096)
- RECORDSIZE(50,50)              -
 NUMBERED)

 -
DATA

 -
 (NAME(EMPLOYEE.KSDS.DATA))        -
 CATALOG(VSAM.USERCAT.TWO)
 /*
```

LDS Definition

```
//LDSMAKE EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD*
 DEFINE CLUSTER                              -
 (NAME(EMPLOYEE.RRDS.CLUSTER) -
VOLUMES(VSAM02)              -
 CYLINDERS(2,1)

-
 LINEAR)

 -
 DATA

 -
 (NAME(EMPLOYEE.KSDS.DATA))        -
 CATALOG(VSAM.USERCAT.TWO)
 /*
```

IDCAMS - REPRO command

It is the general-purpose command that can operate on both VSAM and non-VSAM datasets. It performs three basic functions:

1.It loads an empty VSAM cluster with records. The data and index components for KSDS are built automatically.
2. It creates a backup of a VSAM dataset on a physical sequential dataset and later it can be used for restore/rebuild the VSAM dataset.
3. It merges data from two VSAM datasets.

Command Syntax

```
 REPRO                                                      -
INFILE(DDNAME) | INDATASET(DATASET-NAME)                    -
OUTFILE(DDNAME) | OUTDATASET(DATASET-NAME)                  -
optional parameters
```

INFILE/INDATASET and OUTFILE/OUTDATASET point to input and output datasets.

When loading a KSDS using REPRO, the input data should be first sorted in ascending sequence by the field that will become the primary key in the output dataset. When loading an ESDS the sort step can be eliminated since the records are loaded in entry sequence. For an RRDS, the records are loaded in relative record sequence starting with 1. The dataset should be sorted on the field that correlates to the relative record number.

REPRO Record Selection

REPRO can be used for selective copying.

Where to start REPRO	Where to stop REPRO	Valid For
FROMKEY(REC-KEY)	TOKEY(REC-KEY)	KSDS, ISAM
FROMADDRESS(RAB)	TOADDRESS(RBA)	KSDS,
FROMNUMBER(RRN)	TONUMBER(RRN)	ESDS RRDS
SKIP(number)	COUNT(number)	KSDS , ESDS, RRDS

SKIP refers to the number of input records to skip before beginning to copy and COUNT specifies the number of output records to copy.

The REUSE Parameter With REPRO

The REUSE option of REPRO will logically delete the existing records of a target KSDS, ESDS or RRDS and add new records from the source dataset as if the target dataset were empty. In other words REUSE sets the value of the ending RBA of the Highest Used control area in the data component (HURBA) to zero.

In order to use REUSE with REPRO it is a prerequisite that the target dataset must have been defined with the REUSE option in the DEFINE CLUSTER command.

Merging DATASETS Using REPRO and REPLACE option

The REPRO command can be used for merging two datasets into one. The target dataset can be a non-empty KSDS, ESDS or RRDS.

The input dataset can be any organization if the target dataset is a KSDS or ESDS. However if the target dataset is an RRDS the source dataset has to be an RRDS. KSDS to KSDS is the most common application of this merge technique.

If the REPLACE option is specified as part of the REPRO command then records with duplicate primary keys (for a KSDS) and duplicate relative record numbers (in the case of an RRDS) will be replaced.

IDCAMS - PRINT Command

It is used to print the contents of a dataset. The output can be made available in various formats. This section deals with the PRINT command and its various mutually exclusive options.

```
PRINT INDATASET(data set-name) | INFILE(dd-name)
options
```

PRINT - CHAR/ HEX/DUMP

This specifies the format in which to print.

CHARACTER(CHAR) prints as EBCDIC, 120 characters per line. This format is mostly used for alphanumeric data. Any combination of bits that does not correspond to a printable character will be printed as a period. If length of the record is greater than 120 characters, it is printed in blocks of 120 characters per line.

HEX format prints each character in the dataset as two hexadecimal digits. A maximum of 120 hexadecimal digits are printed on each line, an equivalent of 60 characters.

DUMP format is a combination of the character and hex formats. Each character is printed both as a character and as the corresponding hex representation. Each print line consists of 64 hex digits and the 32 related characters. If the record length is greater than 32 characters the rest of the record is printed in blocks of 64 hex digits and 32 corresponding characters per line.

PRINT – SKIP, COUNT, FROM and TO

The records to be printed can be selected in the same way records are selected in REPRO to COPY.

Where to start Printing	Where to stop Printing	Where Used
SKIP(number)	COUNT(number)	KSDS, ESDS, RRDS, non-VSAM
FROMKEY(key-value)	TOKEY(key-value)	KSDS, ALTERNATE INDEX
FROMADDRESS(rba)	TOADDRESS(rba)	KSDS, ESDS
FROMNUMBER(rrn)	TONUMBER(rrn)	RRDS

A command that prints records 29, 30, and 31 in character format

```
PRINT INDATASET(MM01.CUSTOMER.MASTER) -
    CHARACTER   -
SKIP(28)

    -
COUNT(3)
```

Empty file- check

If the file is empty, PRINT COUNT(1) ends with return-code as 4.

```
PRINT INDATASET(OUTPUT.DSN) CHARACTER COUNT(1)
```

IDCAMS - DELETE Command

The DELETE command can be used to delete both VSAM and non-VSAM objects.

Syntax: DELETE ENTRY NAME OBJECT optional-parameters

OBJECT- CLUSTER GDG PATH AIX ALIAS CATALOG NONVSAM SPACE USERCATALOG

Options – ERASE/NOERASE PURGE/NOPURGE SCRATCH/NOSCRATCH FILE FORCE/NOFORCE

DELETE- ENTRY NAME

The name of the entry to delete is coded. GENERIC NAMES can be also given.

DELETE - ERASE (ERAS) / NOERASE (NERAS)

When the DELETE command is executed it does not physically delete a dataset. Only its ICF or VSAM catalog entry is removed making this space available for future use. The data component remains on the disk until another dataset is allocated on the same spot.

If the DELETE command is executed with the ERASE option, not only will the entry be removed from the catalog but also the data component will immediately be overwritten with binary zeroes. This option is coded for sensitive data files.

DELETE - PURGE(PRG) / NOPURGE (NPRG)

NOPURGE specifies that the entry not to be deleted if the retention period has not expired. (Default). So DELETE command don't delete the dataset before expiry date. PURGE parameter must be coded to delete the dataset before retention period.

DELETE - SCRATCH(SCR) / NOSCRATCH(NSCR)

When a dataset is created, it will be listed in the catalog as well as the VTOC of the DASD. SCRATCH specifies that the dataset to be removed from VTOC as well as catalog. NOSCRATCH specifies that the dataset to be removed from the catalog alone.

DELETE - FORCE(FRC)/NOFORCE(NFRC)

It specifies whether objects that are not empty should be deleted.

FORCE allows you to delete data spaces, generation data groups, and user catalogs without first ensuring that these objects are empty.

NOFORCE causes the DELETE command to terminate when you request the deletion of a data space, generation data group, or catalog that is not empty.

DELTE - FILE(DDNAME)

It specifies the name of the DD statement that identifies:

1. The volume that contains a unique data set to be deleted.
2. The partitioned data set from which a member (or members) is to be deleted.
3. The data set to be deleted if ERASE is specified.
4. The volume that contains the data space to be deleted.
5. The catalog recovery volume(s) when the entry being deleted is in a recoverable catalog. If the volumes are of a different device type, concatenated DD statements must be used. The catalog recovery volume is the volume whose recovery space contains a copy of the entry being deleted.

IDCAMS - LISTCAT Command

The output of this command gives an insight into the inner functioning of VSAM. LISTCAT is used to view dataset attributes, password & security information, usage statistics, space allocation information, creation and expiration dates and much more.

LISTCAT stands for LISTing a CATalog entry. It is useful for listing attributes and characteristics of all VSAM and non-VSAM objects cataloged in a VSAM or ICF catalog. Such objects can be the catalog itself, its aliases, the volumes it owns, clusters, alternate indexes, paths, GDG's, non-VSAM files etc. The listing also provides statistics about a VSAM object from the time of its allocation, namely the number of CI and CA splits, the number of I/O on index and data components, the number of records added, deleted and retrieved besides other useful information.

Syntax:
LISTCAT ENTRIES(OBJECT-NAME) ALL|ALLOCATION|VOLUME|HISTORY|
NAME

Parameter	Meaning
NAME	List the name and type of entry.
HISTORY	Lists reference information for the object including name, type of entry, creation and expiration date and the release of VSAM under which it was created.
VOLUME	Lists the device type and one or more volume serial number of the storage volumes where the dataset resides. HISTORY information is also listed.
ALLOCATION	Lists information that has been specified for space allocation including the unit(cylinders, tracks etc.), number of allocated units of primary and secondary space and actual extents. This is displayed only for data and index component entries. If ALLOCATION is specified VOLUME and HISTORY are included.
ALL	All the above details are listed

LEVEL Parameter in LISTCAT

LISTCAT ENTRIES(SMSXL86.*) ALL
This will return complete information about all objects having two levels of qualification and having the first qualifier SMSXL86. But if we require information about all objects having the high level qualifier SMSXL86, we have to use the LEVEL parameter. LISTCAT LEVEL(SMSXL86.*) ALL

Example

```
        //SMSXL861 JOB (36512),'MUTHU',NOTIFY=&SYSUID
        // LISTCAT EXEC PGM=IDCAMS
        //SYSPRINT DD        SYSOUT=*
        //SYSIN    DD*
 LISTCAT ENTRIES(SMSXL86.PAYROLL.MASTER) -
           GDG   -
 ALL
/*
```

<u>IDCAMS - EXPORT and IMPORT commands</u>

The EXPORT / IMPORT commands can be used for

1. Backup and recovery.
2. Exporting a dataset, an alternate index or a catalog to a different (but compatible) system.

Advantages over REPRO

3. Catalog information is exported along with the data.
4. Cluster deletion and redefinition are not necessary during the import step because input dataset already contains catalog information.
5. Also since the dataset contains catalog information it can be easily ported to other systems. An exported dataset has cross-system portability.

But EXPORT / IMPORT can be used with VSAM datasets only and the dataset created by EXPORT can have only a sequential organization. The dataset created by the EXPORT step is not process-able until it has gone through a corresponding IMPORT step. Thus performing an EXPORT/IMPORT takes up more time.

```
Syntax: EXPORT entry-name OUTFILE(ddname) / OUTDATASET(dsname) -
         optional parameters
```

Entry-name is the name of the object that need to be exported. OUTFILE mention exported into what name.

<u>EXPORT - INHIBITSOURCE|NOINHIBITSOURCE</u>

It specifies whether the original data records (the data records of the source cluster or alternate index) can be accessed for any operation other than retrieval after a copy is exported. This specification can later be altered through the ALTER command.

INHIBITSOURCE (INHS) - cannot be accessed for any operation other than retrieval. NOINHIBITSOURCE - original data records in the original system can be accessed for any kind of operation.

<u>EXPORT - INHIBITTARGET(INHT)/NOINHIBITTARGET (NINHT)</u>

It specifies how the data records copied into the target alternate index or cluster can be accessed after they have been imported to another system. The ALTER command is used to alter this parameter.

INHIBITTARGET - cannot be accessed for any operation other than retrieval.

NOINHIBITTARGET - Target object can be accessed for any type of operation after it has been imported into another system.

<u>EXPORT - TEMPORARY|PERMANENT</u>

It specifies whether the cluster or alternate index to be exported is to be deleted from the original system.

TEMPORARY specifies that the cluster or alternate index is not to be deleted from the original system. The object in the original system is marked as temporary to indicate that another copy exists and that the original copy can be replaced.

PERMANENT specifies that the cluster or alternate index is to be deleted from the original system. Its storage space is freed. If its retention period has not yet expired, you must also code PURGE. PERMANENT is the default.

EXPORT - ERASE|NOERASE

This specifies whether the data component of the cluster or alternate index to be exported is to be erased or not (overwritten with binary zeros).

With ERASE specification, the data component is overwritten with binary zeros when the cluster or alternate index is deleted.

With NOERASE specification, the data component is not overwritten with binary zeros when the cluster or alternate index is deleted.

Example:

```
        //EXPORT EXEC PGM=IDCAMS
        //DD2  DD DSN=SMSXL86.LIB.KSDS.BACKUP(+!),
        //       DISP=(NEW,CATLG,DELETE),UNIT=TAPE,
        //       VOL=SER=121212,LABEL=(1,SL),
        //       DCB=(RECFM=FB,LRECL=80)
//SYSIN DD   *
  EXPORT A2000.LIB.KSDS.CLUSTER         -
   OUTFILE (DD2)
/*
```

IDCAMS - IMPORT command

The IMPORT command restores the dataset from a backup copy created by the EXPORT command. The INFILE parameter specifies the DDNAME that refers to the backup copy created by the EXPORT command.

```
IMPORT INFILE(ddname)/ INDATASET(dsname) -
          OUTFILE(ddname) / OUTDATASET(dsname)               -
           optional parameters
```

IMPORT - INTOEMPTY (IEMPTY)

When this parameter is specified, a previously exported dataset can be imported into an empty previously defined cluster. If this parameter is not specified an attempt to import into an empty dataset will fail.

IMPORT - OBJECTS

When the OBJECTS parameter is coded the attributes of the new target dataset can be changed. These attributes include VOLUMES and KEYRANGES.

IDCAMS-VERIFY Command

If a job terminates abnormally and a VSAM dataset is not closed, the catalog entry for the dataset is flagged to indicate that the dataset may be corrupt(as the end of file or last key is not updated in the index properly). In such case you would get VSAM status code of '97' in the open of the file in the program. Before the dataset can be opened again, the VERIFY command must be used to correctly identify the end of the dataset and reset the catalog entry. Alternate solution is open the file in File-aid in edit mode and just save it. It would update the index with end of file.

Model syntax for the VERIFY command:

```
VERIFY {FILE(ddname[/password]) | DATASET(entryname[/password])}
```

Base cluster and alternate index can be verified. The verification of base cluster does not verify its alternate indexes so each one of them must be treated separately.

IDCAMS - ALTER Command

The ALTER command is used to change many of the attributes of an existing VSAM dataset. It is easier doing it this way than to delete the old dataset and to redefine it with the new attributes. If the latter method is used the data will have to be backed up before deletion to preserve the records in it and restored after reallocation.

It is important to note that all the attributes cannot be changed with ALTER. Few parameters that can’t be altered are: CISZ, Type of Cluster, IMBED/REPLICATE, REUSE/NOREUSE

Syntax: ALTER entry-name parameters

Frequently ALTER is used for renaming (NEWNAME), altering free space (FREESPACE), temporarily disabling updates over dataset (INHIBIT) and extend the generation limit of GDG (LIMIT).

Example

JCL for an AMS job with comments that runs an ALTER command

```
        //SMSXL861 JOB (36512),'Muthu',NOTIFY=&SYSUID
        //NEWNAME EXEC PGM=IDCAMS
        //SYSPRINT DD       SYSOUT=*
        //SYSIN  DD   *
  ALTER MM01.CUSTOMER.MASTER –
      NEWNAME(MM01.CUSTMAST)        /* CHANGE NAME
      */ -
  FREESPACE(10 10)                   /* CHANGE
      FREESPACE ALLOCATION */
/*
```

SYSIN for Adding candidate Volumes

```
         ALTER                          -
        EMPLOYEE.KSDS.DATA              -
        ADDVOLUMES (VSAM03)
```

IDCAMS - MODAL COMMANDS

Functional commands are used to perform function. Modal commands are used to test and SET the condition codes returned by the functional commands.

LASTCC contains the return-code of the last executed functional command and if more than one functional command is executed in single IDCAMS step, then MAXCC contains the maximum value of condition codes from the previously executed functional commands. MAXCC is returned as step return code in JCL.

SET command is used to reset the MAXCC or LASTCC within AMS.

IF-THEN-ELSE statements are used to control the command execution sequence.

In the below example, REPRO command loads data from a sequential dataset on to a KSDS. Only if the condition code of the REPRO step is zero, the next LISTCAT step will be executed. Otherwise the KSDS will be deleted. MAXCC is set to zero at the end to avoid non-zero return code.

```
        //SYSIN        DD *
        REPRO INDATASET(SMSXL86.DATA.TEST) -
 OUTDATASET(SMSXL86.TEST.KSDS)
IF LASTCC = 0                          -
 THEN                                  -
LISTCAT                        -
ENTRIES(SMSXL86.TEST.KSDS) ALL
 ELSE                                          -
DELETE SMSXL86.TEST.KSDS
 END-IF
SET MAXCC=0
/*
```

<u>ICAMS step Return codes and meaning</u>

Error Code	Severity of Error	Meaning
0	No Error	Functional command completed its processing successfully
4	Minor Error	Processing is able to continue, but a minor error occurred, causing a warming message to be issued
8	Major Error	Processing is able to continue, but a more severe error occurred, causing major command specifications to be bypassed
12	Logical Error	Generally, inconsistent parameters are specified, causing the entire command to be bypassed
16	Severe Error	An error of such severity occurred that not only can the command causing the error not be completed, the entire AMS command stream is flushed

<u>ALTERNATE INDEX</u>

<u>Need for AIX</u>

1. KSDS dataset is currently accessed using a primary key. We want to access the dataset RANDOMLY with another key.
2. We want to access ESDS file RANDOMLY based on key.

Steps Involved

Step1. Define AIX.

DEFINE ALTERNATEINDEX command defines an alternate index. Important parameters of this command are:

Parameter	Meaning
RELATE	Relates AIX with base cluster
NONUNIQUE/ UNIQUE	Duplicates are allowed / not allowed in alternate key.
KEYS	Defines the length and offset of alternate key in base cluster
UPGRADE	Adds the AIX cluster to the upgrade set of base cluster. Whenever base is modified, its upgrade set is also modified. UPGRADE is default. NOUPGRADE didn't add the AIX to base cluster upgrade set.
RECORDSIZE	Specifies the record size of alternate index record. It is calculated using the formula in the next table.

AIX Record Size Calculation:

	UNIQUE Alternate Key	NONUNIQUE Alternate Key with maximum n duplicates
KSDS	5+Alternate Key length +Primary Key Length	5+Alternate Key length +(n * Primary Key Length)
ESDS	5+Alternate Key length +RBA size (RBA size = 4 bytes)	5+Alternate Key length +(n * RBA size) (RBA size = 4 bytes)

Five Byte control information of Alternate Index Record:

Byte-1	Type of Cluster; X'00' indicates ESDS and X'01' indicates KSDS.
Byte-2	Length of base cluster pointers in alternate index; Primary key length for KSDS and X'04' for ESDS.
Byte-3 Byte-4	Half word binary indicates number of occurrences of primary key pointers in alternate index record. X'0001' for unique alternate key.
Byte-5	Length of alternate key.

Step2. BLDINDEX

Alternate index should have all the alternate keys with their corresponding primary key pointers. After the AIX is defined, this information should be loaded from base cluster. Then only we can access the records using AIX. BLDINDEX do this LOAD operation.

Important parameters of this command are:

1. INFILE and OUTFILE points to Base Cluster and Alternate index Cluster.

2. INTERNALSORT, EXTERNALSORT, WORKFILES:

Loaded AIX should have sorted alternate record keys with base cluster keys. So an intermediate SORT is required in between reading the base cluster and loading the AIX. We should allocate work datasets IDCUT1 and IDCUT2 for this SORT.

If INTERNALSORT parameter is coded, then AMS tried to do internal sort if enough memory is available and it would be fast. If enough space is not available, then if you have coded IDCUT1 and IDCUT2, it uses that space and does an external sort.

If you code EXTERNALSORT parameter, then AMS wont try for internal sort. If you want to give your own DDNAME files for sorting instead IDCUT1 and IDCUT2, inform it to AMS by WORKFILE(DDNAME1, DDNAME2) parameter. Here, DDNAME1 and DDNAME2 are allocated instead of IDCUT1 and IDCUT2.

Step3. Define Path

The DEFINE PATH command establishes a bridge between the base cluster and an alternate index. It does not occupy any space. It is just a catalog entry, which establishes a link between an alternate index and a base cluster. Important parameters of this command are:

1. PATHENTRY command relates PATH with AIX.
2. When you open a PATH, respective base cluster will automatically be opened. UPDATE opens the UPGRADE set of the base cluster also and it is default. NOUPDATE opens only the base cluster and not its the UPGRADE set.

In BATCH, if you want to access base cluster with alternate key, you should allocate the PATH in the JCL. The name of the DDNAME for the PATH in JCL is DDNAME of the base cluster suffixed with 1 for first alternate key and n for nth alternate key. Refer the COBOL material for more information.

In CICS, if you want access base cluster with alternate key, then you should register the PATH as FCT entry.

Example:

1.Command that defines an alternate index

```
DEFINE AIX    ( NAME(MMA2.EMPMAST.SSN.AIX)            -
                 RELATE(MMA2.EMPMAST)                  -
                  KEYS(9 12)                           -
                 UNIQUEKEY                             -
                  UPGRADE                              -
```

```
                REUSE                                  -
                VOLUMES(MPS800) )                      -
     DATA         ( NAME(MMA2.EMPMAST.SSN.AIX.DATA)
                                    -  CYLINDERS(1 1) )  -
     INDEX ( NAME(MMA2.EMPMAST.SSN.AIX.INDEX) )
```

2.Command that defines PATH

```
DEFINE PATH ( NAME(MMA2.EMPMAST.SSN.PATH)          -
                PATHENTRY(MMA2.EMPMAST.SSN.AIX) -
                UPDATE )
```

3.A job that builds an alternate index

```
//MM01LC3 JOB (36512),'MUTHU',NOTIFY=&SYSUID
//BLDINDX EXEC PGM=IDCAMS
//SYSPRINT DD    SYSOUT=*
//IDCUT1    DD   UNIT=SYSDA,VOL=SER=MPS800,DISP=OLD
//IDCUT2     DD  UNIT=SYSDA,VOL=SER=MPS800,DISP=OLD
//SYSIN       DD    *
 BLDINDEX INDATASET(MMA2.EMPMAST)           -
       OUTDATASET(MMA2.EMPMAST.SSN.AIX) -
       CATALOG(MMA2)
/*
```

The order of build index and definition of PATH
does not matter.

AMP Parameter in JCL

The Access Method Parameter (AMP) completes information in an Access method Control Block (ACB) for a VSAM data set.

The ACB can be coded for a key-sequenced (KSDS), entry-sequenced (ESDS), or relative record (RRDS) VSAM data set. AMP is only supported for VSAM data sets.

AMP is most often used to allocate I/O buffers for the index and data components for optimizing performance.

Syntax: AMP=(sub parameter, sub parameter,)

or
AMP=('sub parameter, sub parameter,')
Single apostrophes are required if the sub parameter(s) contain special characters.

AMORG

This parameter, which stands for Access Method ORGanization, indicates that the particular DD statement refers to a VSAM dataset.

BUFND

This parameter gives the number of I/O buffers needed for the data component of the cluster. The size of each buffer is the size of the data CI.

The default value is two data buffers one of which is used only during CI/CA splits. Therefore the number of data buffers left for normal processing is one.

If more data buffers are allocated, then performance of sequential processing will improve significantly.

BUFNI

This parameter gives the number of I/O buffers needed for the index component of the cluster. Each buffer is the size of the index. This sub-parameter may be coded only for a KSDS because ESDS and RRDS do not have index components. The default value is one index buffer.

If more index buffers are allocated, then performance of random processing will improve significantly.

BUFSP

This parameter indicates the number of bytes for data and index component buffers. If this value is more than the value given in the BUFFERSPACE parameter of the DEFINE CLUSTER, it overrides the BUFFERSPACE. Otherwise BUFFERSPACE takes precedence. The value of BUFSP is calculated as

BUFSP = DATA CISIZE x BUFND + INDEX CISIZE x BUFNI

However it is recommended not to code this parameter and let VSAM perform the calculations from the BUFND and BUFNI values instead.

Other parameters are TRACE, STRNO, RECFM, OPTCD, CROPS.

If SMS is active, then VSAM datasets can be created in JCL without using IDCAMS as below:

```
//KSDSFILE DD DSN=DEVI.CUST.MASTER,DISP=(NEW,CATLG,DELETE),
//                  SPACE=(CYL,(10,10)),
//                  LRECL=100,KEYOFF=10,KEYLEN=12,RECORG=KS
```

RECORG can also be ES(for Entry Sequenced Datasets), RR(for Relative Record datasets) and LS(for Linear Datasets).