21CS VSE^n VSAM Basics

Is it a file or a folder…
it is a VSAM dataset

Edmund Wilhelm
VSE^n Core Senior Software Engineer

Agenda

What is a File System?

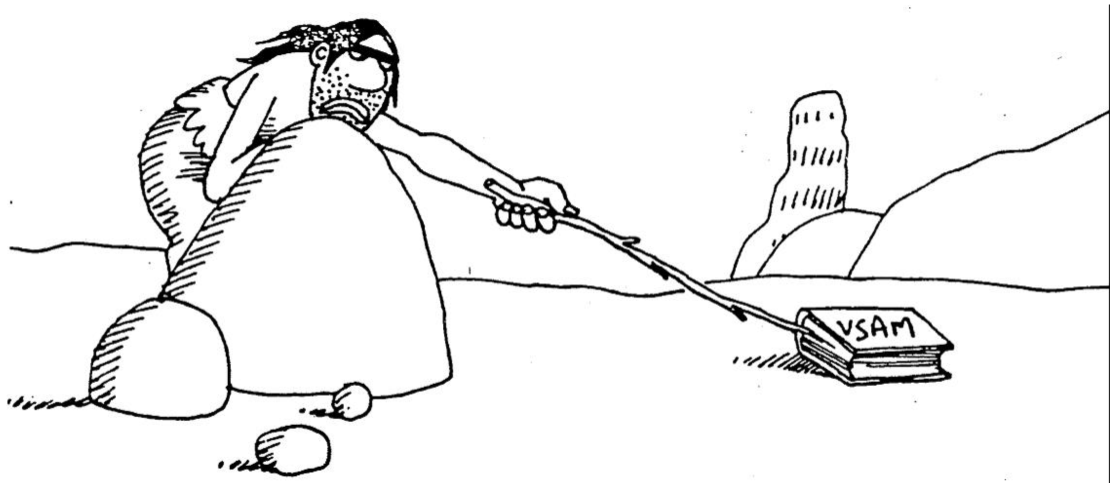
- A method and data structure that the operating system uses to control how data is stored and retrieved.[1]
- Before the advent of computers, the term file system was used to describe a method of storing and retrieving paper documents.[2]

- Consist of at least two layers:
  - Logical file system interaction with the user application, OPEN, CLOSE, READ, etc.
  - Physical file system physical operation of the storage device

Mainframe File System History

- Initially designed to support removable disk packs
- All information regarding files on a disk stored in Volume Table of Contents (VTOC)
- Files on a mainframe are referred to as data sets
- Naming conventions were up to 8 characters separated by dots for a maximum of 44 characters
- To reference a file, it was necessary to provide information about its location (which DASD and where the file started on the DASD) and size. This size was fixed and had to be specified at file creation (i.e., // ASSGN and // EXTENT statements).

Mainframe File System History...



Predecessors:

- SAM – sequential access
- BDAM – direct access
- ISAM – indexed access

All based on the VTOC disk structure.

What is VSAM?

- Virtual Storage Access Method
  - For Direct Access Storage Devices (disk devices)
  - Provides routines which can be used by other programs to transfer data to and from a physical device
  - With VSAM a file can be referred to by name without the user needing to specify where the file is stored, or its size and it can be extended dynamically
  - A VSAM file is called a cluster. Each cluster has one or more components, e.g., a DATA component and an INDEX component

What is VSAM?...

- Space on a DASD must be allocated to VSAM, which it suballocates to files

- Uses a central file called a “catalog” to manage the space and files.

- VSAM has one Master Catalog.

- There can be multiple User Catalogs and there is a pointer in the Master Catalog to each User Catalog

- There can only be one VSAM Catalog per volume

What is VSAM?...

- VSAM catalogs are like a directory within a directory.
- In a DASD VTOC you can see entries that show the space allocated to VSAM with the following format.

  For a data space which can be sub-allocated to VSAM files, the VSAM-generated name is:

  Z999999n.VSAMDSPC.Taaaaaaa.Tbbbbbbb

  where:

n=2	if no catalog resides in the space
n=4	if a user catalog resides in the space
n=6	if the master catalog resides in the space
aaaaaaabbbbbbb	is the time stamp value

What is VSAM?...

General Information

- VSAM is used primarily for application data. It is not used for source programs, JCL, or executable modules. VSAM files cannot be routinely displayed or edited.
- Most VSAM routines are loaded into the Shared Virtual Area (SVA) to be used by all partitions.
- Most of the VSAM functions are available via the Interactive User Interface (IUI).

Functional Areas

- Catalog Management
  - Volume, File information
  - Usage statistics
- DASD Space Management
  - Space allocation, including secondary allocations
  - VSAM and non-VSAM files (VSAM managed SAM, etc.)
  - System Files
  - Libraries
- Open/Close Management
  - Connects and disconnects a cluster with an application program
  - Ensures access integrity

Functional Areas…

- Record Management
  - Performs all I/O access to clusters and catalogs
  - Manage buffer pools
  - Ensure cluster's data integrity

- Utilities
  - IDCAMS (Integrated Data Cluster Access Method Services)
  - IKQVCHK (Catalog Check)
  - IKQVEDA (SNAP Trace)
  - IKQVDU (Maintain VTOC and Labels on Disk)
  - IKQPRED (Compression Prediction Tool)

VSAM Data Organizations – Logical Records

VSAM files are referred to as CLUSTERS.

- Key Sequenced Data Set (KSDS)
  - Indexed, stored logically in key sequence
- Entry Sequenced Data Set (ESDS)
  - Non-indexed, stored in order inserted, new records at end of file
- Relative Record Data Set (RRDS)
  - Non-indexed, stored in relative record order, fixed length records
- Variable Length Relative Record Data Set (VRDS)
  - Indexed, stored in relative record order, but variable length records

VSAM Data Organizations – Logical Records…

Type	Record Length	Sequence by
Entry Sequence Data Set (ESDS)	Fixed or variable	Entry
Key Sequence Data Set (KSDS)	Fixed or variable	Key field
Relative Record Data Set (RRDS)	Fixed only	Record number or entry
Variable-length Relative-record Data Set (VRDS)	Fixed or variable	Record number or entry

VSAM Data Organizations – Logical Records…

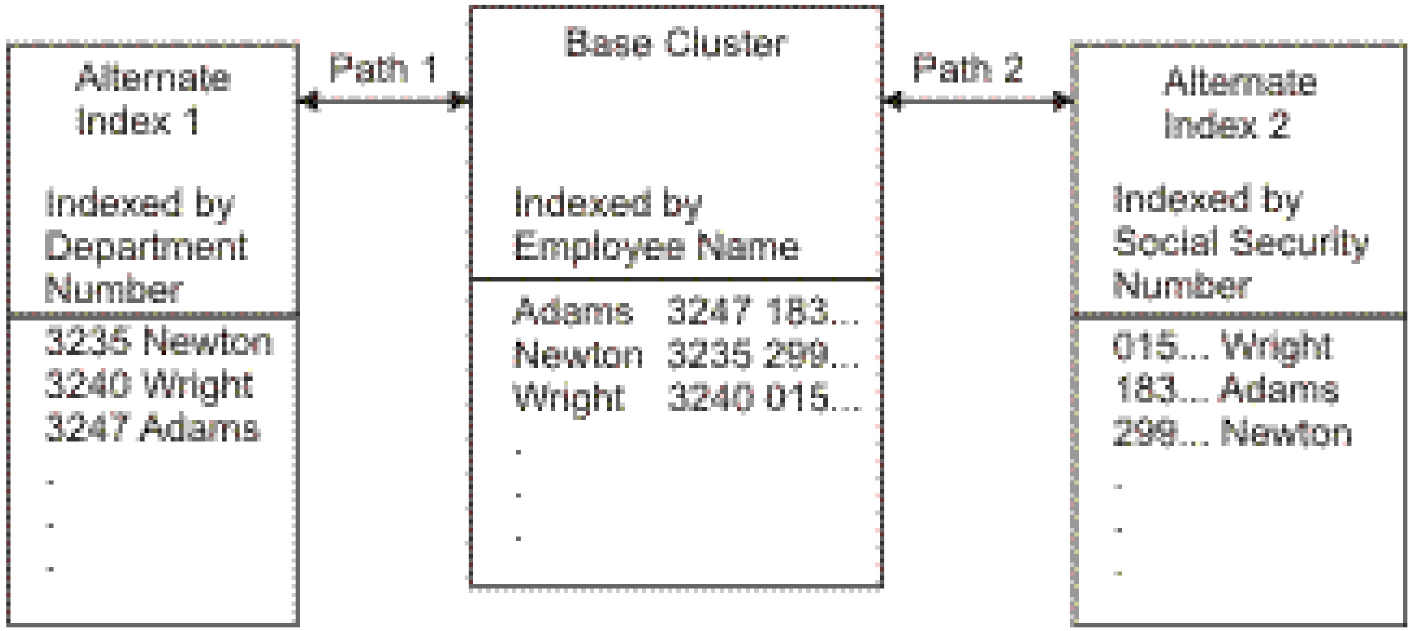
Alternate Index (AIX)

An alternate index provides you with another way of gaining access to the records in a selected KSDS or ESDS file. It eliminates the need for you to keep several copies of the same information organized in different ways for different applications. For example, you can take a KSDS payroll file that is indexed by employee name, and using the same base data, index it according to department number or social security number. You can use any field in the records of the file as an alternate index key field, if the field has a fixed length and fixed position in the record.

VSAM Data Organizations – Logical Records...

Path

To use an alternate index to the file with its prime index (base cluster), you must define a path to it. The path sets up an association between the alternate index and the base cluster.



VSAM Data Organizations – Logical Records…

Entry Sequence Data Set (ESDS)

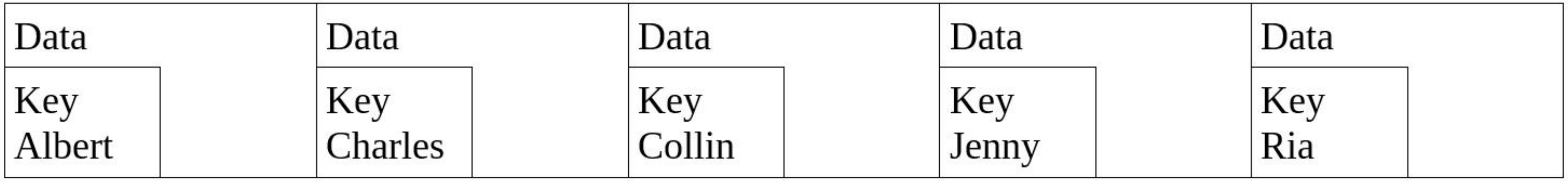
This form of VSAM data keeps records in sequential order. Records can be accessed sequentially.

First Record	Second Record	Third Record	Fourth Record	…

VSAM Data Organizations – Logical Records…

Key Sequence Data Set (KSDS)

This is the most common use for VSAM. Each record has a key field and a record can be retrieved (or inserted) by key value. This provides random access of data. Records are variable length.



VSAM Data Organizations – Logical Records…

Relative Record Data Set (RRDS)

Allows retrieval of records by number; records 1, record 2, and so forth. This provides random access and assumes the application program has a way to derive the desired record numbers.

Relative Record 1		Relative Record 3	Relative Record 4		Relative Record 6

VSAM Data Organizations – Logical Records...

Variable-length Relative-record Data Set (VRDS)

Same as a Relative Record Data Set (RRDS) however with the ability to handle variable length records.

RRN 1	RRN 2	RRN 3	RRN 4	RRN 5
Data	Data	Data	Data	Data

VSAM Data Organizations – Physical Records

Control Interval (CI)

Logical records are in fact physically written in groups of several records on a DASD. This “grouping” of several records together is referred to as blocking. The first block that records are grouped into are what is known as Control Intervals (CI).

CIs are the smallest units of data that may be transferred between a DASD and main storage. A CI size can be from 512 byes to 32 KB.

VSAM Data Organizations – Physical Records…

Control Area (CA)

CIs are again grouped together to what is known as Control Areas (CA). The minimum size of a CA is the size of a track. The maximum size is the size of a whole cylinder – there are 15 tracks per cylinder for (E)CKD devices.

VSAM Data Organizations – Physical Records…

Relative Byte Address (RBA)

In order to know which CI contains a particular record VSAM uses the Relative Byte Address (RBA) of the record. It is the offset of its first byte from the beginning of the data set. VSAM calculates which CI contains the record by dividing the RBA by CI size to get the CI number.

The first record in a VSAM data set has an RBA of zero. The second record has an RBA equal to the length of the first record, and so on. VSAM returns to the application the RBA of the stored record. The RBA of a record in VSAM includes the free space and control information that is stored sequentially before this record. An RBA might change when records are added, deleted, or changed in size.

VSAM Data Organizations – Physical Records…

Relative Byte Address (RBA)…

There are two important RBA values, which are kept in the VSAM catalog:

- High used RBA (HURBA) is the RBA of the first available byte for inclusion at end of the used part of a VSAM data set. HURBA is not rounded to the CA size.

- High allocated RBA (HARBA) is the RBA of the last byte in the allocated part of a VSAM data set. HARBA is rounded to the CA size.

File Identifiers

When you define a new data set (or when the operating system does), you must give the data set a name. This name is referred to as an identifier. This is the name that you see in the Catalog. You specify a name that contains from 1 through 44 alphameric characters, national characters (@, #, and $), and two special characters (the hyphen and the 12-0 overpunch). Names containing more than eight characters must be segmented by periods; one through eight characters may be specified between periods. The first character of any name or name segment must be either an alphabetic or a national character. The last character in the name cannot be a period.

Disk Label Information (DLBL)

A DLBL statement is used to specify a data set used by an application. It links a data set to a symbolic name that is coded in the program. A DLBL statement has many operands.

```
// DLBL filename, 'file_id', date, VSAM, DISP=(d1,d2,d3)
```

The first four are positional and must be coded in a predefined sequence. All operands are optional, except the first one, which specifies the symbolic name.

Disk Label Information (DLBL) Cont.

When VSAM has been specified, additional operands can be specified such as:

- BLK=n|(n,n1)
- BUFDAT=RMODE31
- BUFND=n
- BUFNI=n
- BUFSP=n
- CAT=filename
- CYL=n|(n,n1)
- DISP=(d1)|(d1,d2)|(d1,d2,d3)
- RECORDS=n|(n,n1)
- RECSIZE=n

Disk Label Information (DLBL) Cont.

Disposition processing applies to reusable files only. The default is:

DISP=(OLD,KEEP,KEEP)

where:

OLD is the default when the file is opened.
The first KEEP is the default when the file is *normally* closed.
The second KEEP is the default when the job is *abnormally* ended.

Disk Label Information (DLBL) Cont.

DLBL Statement Disposition Values			
	Possible value	Indicates the file is:	Applies when the file is:
d1	NEW	reset at OPEN	opened
	OLD	not reset at OPEN	
d2	DELETE	made inaccessible at CLOSE	regularly closed
	KEEP	kept at CLOSE	
	DATE	kept at CLOSE, if the expiration date has not been reached	
		made inaccessible at CLOSE, if the expiration date has been reached	
d3	DELETE	made inaccessible at CLOSE	abnormally terminated
	KEEP	kept at CLOSE	

Disk Label Information (DLBL) Cont.

Search Sequence of Catalogs

VSE/VSAM follows a certain order in searching for the catalog for a file. The established hierarchy that determines the specific catalog to be searched is as follows:

1. Explicitly specified user or master catalog.
   This is the catalog that is specified by the IDCAMS CATALOG parameter or by the CAT=filename parameter of a VSAM file's // DLBL statement.

2. Job catalog.
   If the above catalog is not specified, the job catalog (IJSYSUC) specified as the filename of a // DLBL statement is searched.

3. Master catalog.
   If the above catalogs are not specified, the master catalog (IJSYSCT) is searched.

Disk Label Information (DLBL) Cont.

DLBL Specifications and Search Sequence of Catalogs			
// DLBL IJSYSUC specified?	// DLBL CAT=filename	CATALOG Parameter Specified?	Catalog to be Searched
Yes	None	No	Job Catalog
Yes	Filename of User Catalog // DLBL	No	User Catalog
Yes	'IJSYSCT'	No	Master Catalog
Yes	'IJSYSUC'	No	Job Catalog
Yes	None	Yes	CATALOG(catname)
No	None	No	Master Catalog
No	Filename of User Catalog // DLBL	No	User Catalog
No	'IJSYSCT'	No	Master Catalog
No	'IJSYSUC'	No	Master Catalog
No	None	Yes	CATALOG(catname)

Tasks

- Defining Catalogs
- Defining Space on a Volume
- Defining Files
- Volume Ownership
- VTOC Handling
- Transporting Files

Tasks cont.

Defining Catalogs

When you define a catalog under VSAM, VSAM allocates a specific amount of data space from the volume to the catalog. This data space is "owned" by the catalog, and it is managed by VSAM. Other VSAM objects can use this space; that is, you can *suballocate* space.

The VSAM catalog is a key-sequenced file composed of a *data part* and an *index part*. The data part of the catalog consists of:

- *Cluster entries* that describe files.
  Cluster entries contain the information that VSAM requires to properly access a file, verify access authorization (if required), and provide statistics on operations performed on a file.

- *Volume entries* that describe direct-access volumes in terms of the allocation of data spaces and the location of available space.
  Volume entries in a catalog enable VSE/VSAM to keep track of data spaces and free storage areas.

Tasks cont.

Defining Space on a Volume

The space you define will be:

- Identified in the *volume table of contents* (VTOC) of the volume.
- Controlled entirely by the VSE/VSAM catalog in which it is defined.

Tasks cont.

Defining Files

VSAM files (or clusters) are stored in VSAM data spaces. Usually, you first define a data space, then you define the files.

All VSAM files of an installation must be cataloged in a VSAM catalog.

When you define VSAM files, you normally do not need any // ASSGN and // EXTENT statements. This is because VSAM automatically allocates space for the files from existing data spaces.

Tasks cont.

Volume Ownership

A given catalog controls (*owns*) any space that is defined in it. This includes the space in which the catalog resides, and the space occupied by VSAM files defined in it.

The VSAM data space occupied by VSAM files is recorded in the *volume entries* in a catalog. The *ownership* of the volume, and the *use* of VSAM data space on a volume are indicated by label entries in the VTOC of the volume.

VSE/VSAM volume ownership does not affect nonVSAM files that reside on the volume.

Tasks cont.

VTOC Handling

The *ownership* bit in the format-4 VTOC label is set to 1 if the volume contains a VSAM data space. The ownership bit indicates that the volume is owned by one or more catalogs but does not identify the volumes.

VSAM generates *names* for data spaces and enters the names in the VTOC of the applicable volume. You want to be able to recognize the names that relate to VSAM when you list the volume's VTOC, when you reinitialize the volume, or when you dump the volume to a tape.

Tasks cont.

VTOC Handling cont.

The VTOC contains the:

- Name of every VSAM data space on the volume, and
- For unique files, the names for the data and index components of a cluster or alternate index (the format-1 VTOC label is identified with the object's entry name).

The names generated by VSAM have the following format:

Z999999n.VSAMDSPC.Taaaaaaa.Tbbbbbbb

where:

n=2 if no catalog resides in the data space

n=4 if a user catalog resides in the data space

n=6 if the master catalog resides in the data space

aaaaaaabbbbbbb is the time stamp value

Tasks cont.

Transporting Files

- EXPORT and IMPORT
- BACKUP and RESTORE
- IDCAMS REPRO

Modeling

You can use the entry of an already-defined alternate index, catalog, cluster, or path as a model for the definition of another object of the same type. When one entry is used as a model for another, its attributes are copied as the new entry is defined.

Modeling permits you to set your own parameter defaults to override system defaults. Once defaults are established, you need not specify them every time you define new objects. An explicit parameter specification, however, overrides defaults established by you (through modeling) and by the system.

The normal IDCAMS DEFINE CLUSTER or DEFINE ALTERNATEINDEX procedure is greatly simplified by reducing the numbers of parameters required. This in turn can reduce the number of errors that are likely to occur, and the number of parameters to which the user needs to be exposed. At the same time, it permits application- and installation-associated standards.

Modeling cont.

VSAM goes through the following sequence in determining which parameter to use in the definition of a cluster or alternate index.

1. Did you explicitly specify a parameter in the define? If yes, VSAM uses it. (If you explicitly specify a space allocation parameter (CYLINDERS, TRACKS, BLOCKS, or RECORDS) at any level of DEFINE CLUSTER/AIX, the space allocation parameter(s) in your model are ignored.)
2. Did you specify MODEL parameter in the define? If yes, go to step 4, below; VSAM creates a file using the parameters specified in MODEL(entryname).
3. Did you specify the NOALLOCATION parameter with a DEFAULT.MODEL.xxxx in a previous DEFINE command, thereby creating a default model? If yes and the file organization matches the entryname, VSAM uses the parameters specified in the default model.
4. If none of the above apply, VSAM uses the system default (if one exists)

Compression

If a cluster is defined with the COMPRESSED attribute, VSE/VSAM attempts to minimize the external storage needs by compressing each record written to the file. The compression algorithm is compatible with the zSeries hardware compression facility, that is when the processor supports the CMPSC instruction, then this hardware instruction is used to compress or expand data.

Compression cont.

Advantages

- Reduction in DASD space.
- Since records in a compressed file are smaller, the resulting relative byte address (RBA) is smaller. While the maximum size of a VSAM data set (excluding extended-addressed KSDS data sets) is still 4GB (x'FFFFFFFF'), more user data can be stored within a single data set.
- More data can be stored per control interval or buffer. The advantages are:
  - For sequential workloads, a new buffer is required less often. This reduces the number of I/O requests.
  - For random access workloads, the control interval size might be decreased, which in turn might speed up the data transfer time.

Compression cont.

Activation

1. Define the Compression Control Data Set (CCDS)

   While the CCDS is *required* for those catalogs that contain compressed clusters, it is recommended to have a CCDS defined for each catalog.
   Catalogs that were newly defined using the VSE^n/VSAM Interactive Interface already have a CCDS defined for them. Otherwise use the skeleton SKVSAMDC in the ICCF library 59.

2. Define the cluster with the COMPRESSED attribute.

3. Load data into the file.
   It is necessary to use *load mode* since VSAM can determine the dictionary only during the initial load mode. You could use, for example, IDCAMS REPRO to load the data from an existing file to DASD or tape.

Shareoptions

SHAREOPTIONS(value1 value2)

Specifies how a file (data or index component of an alternate index) can be shared within one VSE^n system, or across two or more VSE^n systems.

Files that are shared *across VSE^n systems* must reside on a shared disk device.

Regardless of the share option values you specify:

- If a file is currently open for another program, you cannot delete, reset, or alter the share option value of the file.
- During the initial load of a file, VSAM treats the share option specification as if it were SHAREOPTIONS(1). After the file is loaded and successfully closed, VSAM uses the specified share option value.

Shareoptions cont.

(value1) or *(value1 value2)* specifies the degree of file sharing on one or more VSE^n systems. The values that can be specified are:

value1=1

Specifies that the file being defined can be opened by any number of requests for input processing.

Once the file is opened for input, it cannot be opened for output processing. Conversely, once the file has been opened for output processing, no other request can open it for input or output until the first output processing has been completed. Share option 1 ensures full read and write integrity.

Shareoptions cont.

value1=2

Specifies that the file being defined can be opened by any number of requests for input processing, even if one request is using it for output processing.

Only one request can open the file for output at one time (every ACB counts as one request). Share option 2 ensures write integrity only (because the file might be modified while records are being retrieved from it). Therefore, read integrity of the file must be ensured by every user.

Shareoptions cont.

value1=3

Specifies that the file can be opened by any number of requests for both input and output processing.

Except for initial load processing, VSAM does nothing to ensure read or write integrity. VSAM, however, ensures that the opened file is not deleted or reset and that its share option value is not altered.

Share option 3 provides good performance, but at the expense of data integrity. The option is intended for users who provide their own read and write integrity.

You should not specify this option if the file is accessed by multiple requests and if you do not provide your own integrity checking. Too many errors can occur that VSAM is unable to detect or correct.

Shareoptions cont.

value1=4 value2≠4

The file can be opened by any number of requests (ACBs) for input processing. At the same time the file can be opened by one or more requests on a single system for output processing. The system that gains exclusive control of the file for output processing will be the one that first issues a request for output processing.

VSAM ensures *write integrity* every time a record is updated or inserted as in share option 2. You can ensure *read integrity* by retrieving records with the *update* option. If you do not use the update option, some records in CIs being updated concurrently by several requests might be missed or skipped by VSAM, because every request might retrieve a different copy of the CI.

Shareoptions cont.

value1=4 value2=4

The file can be opened by any number of requests (ACBs) for output processing from multiple systems.

VSAM ensures write integrity every time a record is updated or inserted (as with share option 2). Read integrity of the file must be ensured by every user.

If you specify SHAREOPTIONS(4 4), read integrity is the same as with SHAREOPTIONS(4).

Note: With SHAREOPTIONS(4 4) specified, the lock-file activity (regarding z/VSE DASD sharing) increases. This may have a performance impact.

IDCAMS

Used to create and maintain VSAM space, catalogs and files. It has many functions:

- Establish catalog(s)
- Create data spaces
- Create VSAM files and load records into the files
- Build an alternate index for a file
- Create backup copies of files and their associated catalog entries
- Print, copy, or reorganize files
- Delete files, data spaces, and catalogs
- Alter file definitions and file attributes
- Print catalog entries

IDCAMS cont.

- Move catalogs and files from one system to another
- Convert nonVSAM files to VSAM files
- Map a VSAM cluster to a relational structure, and later maintain the associated map or view
- Recover from damage to files or catalogs
- Copy entire volumes to support offline backup to tape from the target volume, for example
- Verify command syntax
- Merge two VSAM files

IDCAMS cont.

All IDCAMS commands have this general structure:

```
VERB parameter(s) terminator
```

VERB specifies the type of service requested. Parameter(s) further describe the service requested. Terminator indicates the end of the command. You can abbreviate many of the verbs and parameters. Permitted abbreviations are listed in the description for every command or parameter.

IDCAMS cont.

Verbs

DEFINE MASTERCATALOG
DEFINE USERCATALOG
DEFINE SPACE
DEFINE CLUSTER
DEFINE ALTERNATEINDEX
DEFINE PATH
BLDINDEX
ALTER
DELETE

EXPORT
IMPORT
REPRO
BACKUP/RESTORE
SNAP
LISTCAT
PRINT
VERIFY

Macros

Declarative VSE/VSAM macros:

• ACB	specifies the file to be processed and the access type.
• EXLST	specifies a list of user-supplied exit routines.
• RPL	specifies information for a particular request.

Macros to Share Resources Between Several Files:

BLDVRP	builds a VSE^{n}/VSAM pool of buffers, control blocks, and channel programs.
DLVRP	deletes a resource pool.
WRTBFR	writes waiting buffer contents to satisfy a GET request.

Macros cont.

Request Macros:

GET	retrieves a record from a file for processing.
PUT	inserts a record in a file.
POINT	positions control to a specific address in the file.
ERASE	deletes a record in a file.
ENDREQ	ends processing of a GET or POINT request.

Macros cont.

Control Block Manipulation Macros:

GENCB	specifies declarative parameters during program execution.
MODCB	changes declarative parameters.
SHOWCB	displays declarative parameters in effect.
TESTCB	checks declarative parameters (or their error codes) and sets the condition code accordingly.
SHOWCAT	retrieves information about files out of a catalog.

Macros cont.

OPEN/CLOSE Macros:

OPEN	connects a program to a file.
CLOSE	prepares the separation and disconnects a program from a file.
TCLOSE	prepares the separation, but leaves program and file connected.

Local Shared Resource (LSR) Pools

- Shared Pool Concept
- Automatic Definition
- Advantages:
  - Better Storage Management
  - Data In Memory
- Monitor LSR Pools
- Index Sub-pools
- Initial Allocation:
  - “Deal the files out”.
- Keep AIX and base clusters in same pool: (OLTP will warn you during start-up).

Dataset Name Sharing (DSN)

- Multiple output OPENs for SHR(2) file
- Shares more than just buffers
- CEDA DEFINE FILE DSNSHARING(ALLREQS)
  -or-
  DFHFCT TYPE=FILE,DSNSHR=ALL,BASE=NAME
- Advantages:
  - Multiple Output OPENs (see above)
  - More efficient usage of storage
  - Buffer sharing, reduce chance of read integrity problems

Dataset Name Sharing (DSN) cont.

- Restrictions:
  - First FCT entry opened determines access mode
  - First FCT entry opened must specify sufficient resources (strings, buffers).
    Recommend: Use LSR.
  - Reusable files are not supported.
- Recommendations:
  - Only use if necessary
  - Use both Dataset Name Sharing and LSR

Limitations

1. Maximum of 123 volumes for a file. For implicitly defined files, VSAM will extract up to 16 volumes from the default model.

2. 4-byte RBA (Relative Byte Address). Every record in a VSAM file is ultimately accessed using its relative byte position from the beginning of the file. This means that you can only have 4.3 billion (4,294,967,295) bytes of data in a single VSAM file. This amounts to about ½ 3390 model 9 dedicated to a single VSAM file. Since VSAM file compression occurs prior to the insertion of records into the control interval, usage of compressed files (introduced in VSE/ESA 2.1), substantially increases the amount of user data that can be "packed" into 4GB. The actual limit is 1 CA less than 4GB.

   With VSE/ESA 2.3, a cluster can be defined as 'ExtraLargeDataset' or 'XXL', which allows up to 289GB for an extended KSDS dataset.

Limitations cont.

3. VSAM files are limited to 123 extents. This is normally only a problem if you specify a very small secondary extent relative to the primary extent. Remember, VSAM will sub-allocate an extent up to 5 times. So, if you ask for 500 cylinders as a primary extent, and VSAM cannot find that much contiguous space, it will attempt to give the file up to 5 extents of 100 cylinders each (or some combination thereof). This may quickly exceed the extent limit for the file. Unique files and reusable files are limited to 16 extents.
4. The lock used by Shareoption(4) processing has a two-byte field for the control area id (or index control interval id for KSDS files). Thus, a Shareoption(4) file may not have more than 64511 (64K-1024) control areas, and the maximum file size would be 64511 times your control area size.

   If you use the maximum control area size for your file of one cylinder, define the file as XXL, and reside on 3380, with a CISIZE of 4K, this will set a maximum file size of 36.9GB.

Limitations cont.

5. Any single allocation (primary or secondary) cannot contain more than 16Meg records, if the user intends on using Backup/Restore.

6. When defining files using RECORDS(..), you can only specify up to 16 million records per extent. You can request more data by specifying a larger maximum recordsize for the file. Remember, if the file is compressed, VSAM uses the un-compressed maximum record length to calculate how much space to reserve for the file. This may give you more space than you need.

7. With z/VSE 4.1, a VSAM catalog can be defined with a primary / secondary allocation of up to 4999 cylinders or 8,288,095 blocks.

Note: customers will start running into severe performance degradation before they reach the outer limit of VSAM file size (merely from the number of index levels you have to run through to find a single record).

Large DASD Support

VSAM supports DASD with a capacity exceeding 65535 (64K) tracks, referred to as *Large DASD*. (Accordingly, DASD with a capacity of 64K tracks or less is referred to as *Small DASD*.) This support provides a capacity of up to 10017 cylinders (150255 tracks), which corresponds to the capacity of an IBM 3390 Model 9.

VSAM recognizes two types of ECKD DASD:

- Small DASD have less than 64K tracks per volume and are reported in a LISTCAT as "3390".
- Large DASD:
  - BIG DASD, have a capacity of more than 64K tracks and are reported in a LISTCAT as "BIG-3390". VSAM supports up to 10017 cylinders on this device.
  - FAT DASD supports a device with more than 64K tracks up to 64K cylinders. LISTCAT reports this device as "FAT-3390".

Large DASD Support cont.

If you try to define a VSE/VSAM catalog or space on a DASD volume that exceeds the limit of 10017 cylinders without the parameter FATDASD, then you will receive the following message:

```
IDC0055I VOLUME SPACE EXCEEDS MAXIMUM VSAM CAPABILITY. MAXIMUM WILL BE USED.
```

Utilities

IKQVCHK – VSAM Catalog Checker:

```
// EXEC IKQVCHK,SIZE=AUTO,PARM='<catalog.name>’
```

- Verifies User Catalog for internal consistency.
- Should be run as part of regularly scheduled maintenance.

IKQVDU – VTOC Management

```
// ASSGN SYS000,DISK,VOL=<volid>,SHR
// UPSI nn
// EXEC IKQVDU,SIZE=AUTO
```

- Manipulates VTOC (delete, define, reset …)

IKQPRED – Compression Prediction Tool

```
// EXEC IKQPRED,PARM=’<catalog.name> / <cluster.name>‘
```

- Checks an entire catalog or a series of files (generic specification supported).

IDCONS

- Interactive console interface to IDCAMS. Useful in system recovery situations.

Conclusion

Thank you for your attention ;-)

References

1. https://tldp.org/LDP/sag/html/filesystems.html
   Linux System Administrators Guide: Chapter 5. Using Disks and Other Storage Media. The Linux Document Project "A *filesystem* is the methods and data structures that an operating system uses to keep track of files on a disk or partition; that is, the way the files are organized on the disk."
2. https://archive.org/details/officepracticea01mcgigoog
   McGill, Florence E. (1922). Office Practice and Business Procedure. Gregg Publishing Company. p. 197. Retrieved August 1, 2016.