

GC26-3799-1

Systems

**OS/VS Virtual Storage
Access Method (VSAM)
Planning Guide**

IBM

Systems

**OS/VS Virtual Storage
Access Method (VSAM)
Planning Guide**

IBM

Second Edition (January 1973)

This edition replaces the previous edition (numbered GC26-3799-0) and makes that edition obsolete.

This edition applies both to Release 2 of OS/VS1 and to Release 1 of OS/VS2, and to all subsequent releases of either system unless otherwise indicated in new editions or technical newsletters.

Each technical change is marked by a vertical line to the left of the change.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/360 and System/370 Bibliography*, GA22-6822, the latest *IBM System/370 Advanced Function Bibliography*, GC20-1763, and the technical newsletters that amend those bibliographies, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch-office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, Programming Center—Publishing, Department D58, Monterey and Cottle Roads, San Jose, California 95193. All comments become the property of IBM.

USING THIS PUBLICATION

This publication is intended to enable prospective users of VSAM (virtual storage access method), an access method of OS/VS (operating system/virtual storage), to prepare for using VSAM. The intended audience is data-processing managers whose decisions will influence the use of VSAM, system and application programmers who will make detailed preparations, and others seeking an introduction to VSAM.

This planning guide has six chapters:

- “Introducing VSAM,” which outlines how VSAM meets the requirements of an access method in today’s data-processing environment.
- “Getting to Know What VSAM Is and Does,” which explains the concepts and functions of VSAM and is required reading for the following chapters.
- “Communicating with VSAM,” which discusses, primarily for programmers, the multifunction service program Access Method Services, the macros of VSAM, and the use of JCL (job control language) with VSAM.
- “Preparing for VSAM,” which indicates, for the planners, the programming languages and optional features of OS/VS that VSAM can be used with.
- “Optimizing the Performance of VSAM,” which outlines, for application and system programmers, ways to achieve the best throughput of which VSAM is capable.
- “Protecting Data with VSAM,” which describes, for managers and system programmers, VSAM’s standard and optional features for data integrity and security.

This publication also has a glossary and an index.

The reader is expected to be familiar with basic concepts such as access method, direct-access storage, and the distinction between data-set organization and data-set processing. The sections dealing with those concepts in *OS/VS Data Management Services Guide*, GC26-3783, are suitable for preparatory reading.

In the chapter “Preparing for VSAM,” the section “How Can Existing Programs That Use ISAM Be Used with VSAM?” is intended for those who use ISAM (indexed sequential access method). Other readers may ignore this section and any other references to ISAM. The section of the *Data Management Services Guide* that discusses ISAM is suitable for reference.

The discussion on JCL in the chapter “Communicating with VSAM” presupposes the reader’s familiarity with the sections of *OS/VS JCL Reference*, GC28-0618, that discuss the JCL parameters described in this planning guide.

Other publications referred to in this publication are:

- *Introduction to Virtual Storage in System/370*, GR20-4260
- *OS/VS Checkpoint/Restart*, GC26-3784
- *OS/VS Master Index*, GC28-0602
- *OS/VS Service Aids*, GC28-0633

- *OS/VS System Management Facilities (SMF)*, GC35-0004
- *OS/VS2 TSO Terminal User's Guide*, GC28-0645
- *OS/VS2 TSO Command Language Reference*, GC28-0646

CONTENTS

Using This Publication	3
Figures	9
Introducing VSAM	11
What is VSAM?	11
What Are the Requirements for an Access Method?	12
How Does VSAM Meet These Requirements?	12
High Performance	12
Applicability to Different Types of Processing	13
Simplicity of Use	13
Protection of Data	13
Central Control	14
Portability of Data Between Systems	14
Device Independence	14
Ease of Conversion	14
What Machines Can VSAM Be Used With?	14
Getting to Know What VSAM Is and Does	15
What Are VSAM's Two Types of Data Sets?	15
The Use of Control Intervals	15
The Control Interval in Perspective	16
The Method of Storing a Record in a Control Interval	17
Key-Sequenced and Entry-Sequenced Data Sets	18
Key-Sequenced Data Sets	18
Entry-Sequenced Data Sets	20
How Are VSAM Data Sets Created?	21
In What Ways Can VSAM Data Sets Be Processed?	21
Keyed Access for Key-Sequenced Data Sets	22
Keyed Sequential Retrieval and Key Direct Retrieval	22
Keyed Sequential Deletion and Keyed Direct Deletion	23
Keyed Sequential Storage and Keyed Direct Storage	23
The Use of Free Space for Processing a Key-Sequenced Data Set	23
Addressed Access for Both Types of Data Sets	25
Addressed Sequential Retrieval and Addressed Direct Retrieval	25
Addressed Sequential Deletion and Addressed Direct Deletion	25
Addressed Sequential Storage and Addressed Direct Storage	25
What Are the Master Catalog and User Catalogs For?	26
A VSAM Catalog's Use in Data and Space Management	26
Information Contained in the Records of a Catalog	27
Information in a Data-Set Record	28
Information in a Volume Record	28
The Special Uses of User Catalogs	28
Improving Performance and Reliability	28
Moving Volumes from One Operating System to Another	29
Communicating with VSAM	31
How Is Access Method Services Used?	31
Defining and Deleting Data Sets and Listing Catalog Entries	31
DEFINE: Defining a Data Set and Allocating Space	32
ALTER: Modifying a Catalog Entry	32
DELETE: Removing a Catalog Entry and Freeing Space	33
LISTCAT: Listing Catalog Entries	33
Copying and Listing Data Sets	33

REPRO: Converting and Reorganizing Data Sets	33
PRINT: Listing Data Records	33
Moving Data Sets from One Operating System to Another	34
EXPORT: Extracting Catalog Information and Making a Data Set Portable	34
IMPORT: Loading a Portable Data Set and Its Catalog Information	34
Recovering from Damage to Data	34
VERIFY: Testing and Reestablishing a Data Set's Integrity	36
What Are the Macros for Processing a VSAM Data Set?	36
Connecting and Disconnecting a Processing Program and a Data Set	36
OPEN: Connecting a Processing Program to a Data Set	36
CLOSE: Disconnecting a Processing Program from a Data Set	36
Specifying Parameters That Relate the Program and the Data	37
ACB: Defining the Access-Method Control Block	37
EXLST: Defining the Exit List	38
RPL: Defining the Request Parameter List	38
GENCB: Generating Control Blocks and Lists	40
Manipulating the Information Relating the Program and the Data	40
MODCB: Modifying the Contents of Control Blocks and Lists	40
SHOWCB: Displaying Fields of Control Blocks and Lists	40
TESTCB: Testing the Contents of Control Blocks and Lists	40
Requesting Access To a Data Set	40
How is JCL Used?	41
Defining a VSAM Data Set	41
Processing a VSAM Data Set	41
Specifying VSAM Catalogs	41
Using Other JCL Parameters	42
JCL Parameters Not Used with VSAM	42
VSAM's Special DD Parameter: AMP	42
Preparing for VSAM	43
What Programming Languages Can VSAM Be Used With?	43
How Can the Time Sharing Option (TSO) Be Used with VSAM?	43
How Can System Management Facilities (SMF) Be Used with VSAM?	44
How Can Existing Programs That Use ISAM Be Used with VSAM?	44
Comparison of VSAM and ISAM	44
Comparison of VSAM and ISAM in Common Areas	44
VSAM Functions That Go Beyond ISAM	45
How to Convert an Indexed Sequential Data Set to a Key-Sequenced Data Set	46
What the ISAM Interface Does	47
Restrictions in the Use of the ISAM Interface	47
Optimizing the Performance of VSAM	49
How Can Control-Interval Size Be Used to Influence Performance?	49
How Does Distributed Free Space Improve Performance	49
What Index Options Are There to Improve Performance?	50
Index-Set Records in Virtual Storage	50
Index and Data Set on Separate Volumes	50
Sequence-Set Records Adjacent to the Data Set	51
Replication of Index Records	51
How Can VSAM Catalogs Affect Performance?	52
Searching a VSAM Catalog	52
Sharing Services with User Catalogs	52

Protecting Data with VSAM	53
How Does VSAM Achieve Data Integrity?	53
Method of Inserting Records into a Key-Sequenced Data Set	53
Control-Interval Principle	53
Method of Indicating the End of a Data Set	54
Preformatting a Data Set	54
Updating the Catalog	54
Verifying Write Operations	54
Protecting the Catalog	55
How Is Shared Data Protected?	55
Subtask Sharing	56
Cross-Region Sharing	56
Cross-System Sharing	57
How Can Passwords Be Used to Authorize Access?	57
How Are Programs Restarted Following a Failure?	58
Recording Checkpoint Information	58
Restarting the Processing Program	58
Restrictions and Options for Restarting a Program	58
How Can the Causes of Problems Be Determined?	59
Exists to Your Error-Analysis Routines	59
VSAM Messages	59
Generalized Trace Facility (GTF)	59
Glossary	61
Index	63

FIGURES

Figure 1.	Position of VSAM Between a Processing Program and Auxiliary Storage	11
Figure 2.	Control Intervals Are Independent of Physical Record Size	16
Figure 3.	Relationship Among Storage Volumes, Data Spaces, and Data Sets	17
Figure 4.	Placement of Data Records and Control Information in a Control Interval	17
Figure 5.	Comparison of a Key-Sequenced and Entry-Sequenced Data Set	18
Figure 6.	Relationship Among the Levels of an Index and a Data Set	19
Figure 7.	Distribution of Free Space in a Key-Sequenced Data Set	20
Figure 8.	Splitting a Control Interval for Record Insertion	24
Figure 9.	Cataloging VSAM and NonVSAM Data Sets in a VSAM Catalog	27
Figure 10.	Comparison of Volume Portability and Data-Set Portability	35
Figure 11.	Use of ISAM Programs to Process VSAM Data Sets	46
Figure 12.	Replication of a Sequence-Set Index Record Adjacent to Its Control Area	52

INTRODUCING VSAM

This chapter is intended for all readers new to VSAM (virtual storage access method). It introduces VSAM, outlines the access-method capabilities that are required in today's data-processing environment, shows how VSAM has those capabilities by describing its area of applicability and summarizing its basic features, and indicates the CPUs (central processing units) and auxiliary-storage devices that VSAM can be used with.

What Is VSAM?

VSAM is a high-performance access method of OS/VS (operating system/virtual storage), option 1 or 2, for use with direct-access storage.

VSAM resides in virtual storage along with the processing program using it. Figure 1 illustrates VSAM's position between the program and the data stored on a direct-access storage device.

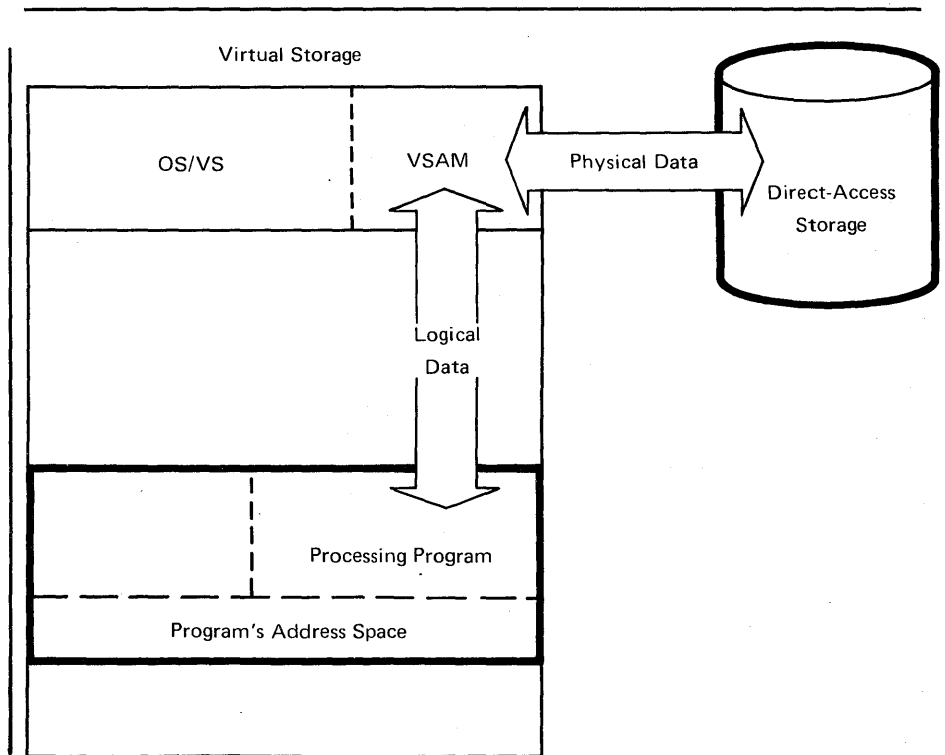


Figure 1. Position of VSAM Between a Processing Program and Auxiliary Storage

What Are the Requirements for an Access Method?

In data processing today, it is common for a computer installation to do a number of different types of processing. An installation must provide for one combination or another of data-base processing, online processing, batch processing, inquiry and transaction processing, communications, and multiple CPUs under the control of different operating systems. This variety requires an access method that provides:

- High performance of retrieval and storage—independent of previous insertions of records into data sets and uninterrupted by requirements to reorganize data sets or copy them for backup
- Applicability to different types of processing that require different kinds of access and different levels of performance (such as online and batch processing)
- Simplicity of use by means of a common set of instructions for different types of access, simplified JCL (job control language), and optimization of the use of space in auxiliary storage
- Protection of data: security against unauthorized access and integrity through prevention of intentional or accidental loss of data
- Central control over the creation, access, and deletion of data sets and over the management of space by keeping data-set and storage information in one place and making it independent of JCL and processing programs
- Ability to move data from one operating system to another in a format that is common to both systems
- Independence from type of storage device: freedom from physical record size, control information, and record deblocking
- Ease of conversion of data and programs from other access methods to the new access method

How Does VSAM Meet These Requirements?

VSAM provides an approach to meeting these requirements through:

- A format for storing data independently of type of storage device
- Routines for sequential or direct access and for access by key or by relative address
- Options for optimizing performance
- A comprehensive catalog for defining data sets and auxiliary-storage space
- A multifunction service program (Access Method Services) for setting up catalog records and maintaining data sets

High Performance

VSAM's high performance is due to an efficiently organized index, performance options for reducing disk-arm movement and rotational delay, and distributed free space for fast insertion of records and minimal reorganization. The size of the index is reduced by compressing keys to eliminate redundant information. The type of index used for a data set is also used for VSAM catalogs to give fast catalog access.

VSAM's method of inserting records into a data set provides access whose

speed following a large number of insertions is equivalent to the speed of access without previous insertions. Free space is used for efficient automatic reorganization of data sets: inserted records are stored and addressed in the same way as original records, and space given up by deletions is reclaimed as free space.

Applicability of Different Types of Processing

VSAM is designed to meet most of the common data-organization needs of both batch and online processing. Batch processing requires the efficiency of sequential and indexed data; online processing requires efficient direct access for random requests. VSAM permits both direct and sequential access access can be by key or by relative address. Different types of processing can be intermixed in the processing of a common data base. You can select the type of access or the combination of types that best suits your application.

You can use VSAM with OS/VS's TSO (Time Sharing Option) for conversational time sharing from remote terminals.

Simplicity of Use

There is a common way of requesting the different types of access (sequential or direct, by key or by relative address), so that the same instructions are learned and used for achieving different results.

All VSAM data sets are cataloged, so JCL is simplified. Minimal JCL parameters are required for describing data sets.

VSAM automatically calculates the optimum-sized units in which to store data. It optimizes the use of virtual-storage space for I/O (input/output) buffers. Programmers can think in terms of the application, not in terms of the internal workings of VSAM.

Individual data records are passed to a processing program without any system control information: application data alone is processed by the program. Application programmers do not need to know the format of control blocks. They need not be concerned either with storage devices and device addresses or with different formats for fixed-length or variable-length records.

Protection of Data

VSAM protects data by means of its design and its integrity and security options. *Integrity* means the safety of data from inadvertent destruction or alteration; *security* means the protection of data from unauthorized use or purposeful destruction or alteration. VSAM writes records in a way that does not expose data to loss, even if an I/O error occurs. You can specify optional passwords for levels of protection (read-only, update, and full access) and include your own routine to check a requester's authority to gain access to data. You can select options for formatting data sets before data is stored in them and for verifying write operations for data integrity. VSAM also provides various levels of protection for data to be shared between subtasks, regions, and operating systems. See the chapter "Protecting Data with VSAM" for additional information.

Central Control

The VSAM catalog brings together extensive information about data sets and storage space. Access Method Services controls the definition and deletion of data sets and the alteration of information about them in the catalog. Its use is authorized by passwords assigned to the data sets or to the catalog itself. Consequently, the management of your inventory of data sets is centralized and made independent of the use of JCL or the actions of processing programs. Space for data sets can be allocated or deallocated without mounting volumes, because the information describing the contents of VSAM spaces on those volumes is contained in the catalog. You can assign a data set to volumes by ranges of keys that are controlled by the catalog.

Portability of Data Between Systems

VSAM's technique for storing records uses a format that is common to OS/VS and DOS/VS (disk operating system/virtual storage). Communication with VSAM is very similar for both operating systems, except for JCL. Access Method Services includes functions that facilitate moving data sets and volumes from one operating system to another.

Device Independence

VSAM is independent of particular types of storage devices, because it addresses a record in a data set without respect to the physical attributes of auxiliary storage, but with respect to the displacement of the record from the beginning of the data set. The unit in which data is transmitted between virtual and auxiliary storage does not depend on the size of the physical records in which data is stored physically on a volume.

Ease of Conversion

VSAM provides for easy conversion of indexed sequential data sets to VSAM format and the continued use of your existing ISAM (indexed sequential access method) programs to process converted data sets and new VSAM data sets. Access Method Services converts a sequential or an indexed sequential data set to VSAM format. To process the converted data set with the ISAM program, a set of interface routines within VSAM interpret each ISAM request and issue the appropriate VSAM request.

What Machines Can VSAM Be Used With?

You can use VSAM on IBM System/370 CPU Models 135 (OS/VS1 only), 145, 155 (Model 2), 158, 165 (Model 2), and 168. Each of these CPUs must have the dynamic address translator that is required by OS/VS1 and OS/VS2. VSAM is designed to take full advantage of the benefits of virtual storage. See the *Introduction to Virtual Storage in System/370*, GR20-4260, for a discussion of virtual storage.

The IBM direct-access storage devices that you can use are the IBM 2314 Direct Access Storage Facility, the 2319 and the 3330 Disk Storage, the 2305, Models 1 and 2, Fixed Head Storage, and the 3333 Disk Storage and Control.

GETTING TO KNOW WHAT VSAM IS AND DOES

Familiarity with the VSAM concepts and terminology introduced in this chapter is presupposed in the following chapters. The concepts are especially important for application programmers who will design and code programs to process data with VSAM, and to system programmers who will maintain the VSAM installation.

This chapter explains the two types of VSAM data sets, discusses creating VSAM data sets and gaining access to them, and describes the master catalog and user catalogs.

What Are VSAM's Two Types of Data Sets?

VSAM has key-sequenced and entry-sequenced data sets. The primary difference between the two is the sequence in which data records are stored in them.

Records are loaded into a *key-sequenced data set* in key sequence: that is, in the order defined by the collating sequence of the contents of the key field in each of the records. Each record has a unique value in the key field, such as employee number or invoice number. VSAM uses an index and optional free space to insert a new record into the data set in key sequence.

Records are loaded into an *entry-sequenced data set* without respect to the contents of the records. Their sequence is determined by the order in which they are physically arranged in the data set: their entry sequence. New records are stored at the end of the data set.

VSAM stores the records of each type of data set in the same way in a fixed-length area of auxiliary storage called a *control interval*. We can better discuss the two types of data sets if we first look at the control interval in perspective with the other logical divisions of a data set and see how and why VSAM uses it for storing records.

The Use of Control Intervals

A control interval is a continuous area of auxiliary storage that VSAM uses for storing data records and control information describing them. It is the unit of information that VSAM transfers between virtual and auxiliary storage. Its size may vary from one data set to another, but for a given data set the size of each control interval in it is fixed, either by VSAM or by you, within limits acceptable to VSAM. VSAM chooses the size that is optimal for the type of direct-access storage device used to store the data set, depending on the size of your data records and the smallest amount of virtual-storage space your processing program will provide for VSAM's I/O buffers.

A control interval is independent of particular types of storage devices. For instance, a control interval that fits on a track of one type of device might span several tracks if the data set were moved to another type of device, as Figure 2 illustrates.

The Control Interval in Perspective

How does a data set relate to the physical attributes of auxiliary storage? And how does a control interval relate to a data set?

A volume can contain areas for VSAM's use and areas for the use of other access methods or the operating system. A storage area defined in the volume table of contents for VSAM's exclusive use is called a *data space*. It can be extended beyond its original size to include up to 16 continuous areas (*extents*) that need not be adjacent to one another on the volume.

A data set is stored in a data space or data spaces on one or more volumes on direct-access devices of the same type. When you define a data set, you can allocate enough space to have some left at the end of the data set for additions. Otherwise, when additional space is needed, VSAM automatically extends the data set by the amount of space indicated in the definition of the data set in the catalog. It can be extended beyond its original size to include up to 255 extents, or to a maximum size of 2^{32} (approximately 4,290,000,000) bytes. Figure 3 illustrates the relationships among volumes, data spaces, and data sets. The figure shows portions of data sets A and C stored in different data spaces on different volumes.

A data set is made up of control intervals. A group of control intervals makes up a *control area*. It is the unit of a data set that VSAM preformats for data integrity as records are added to the data set. (See the section "Method of Indicating the End of a Data Set" in the chapter "Protecting Data with VSAM.") In a key-sequenced data set, control areas are also used for distributing free space throughout the data set as a percent of control intervals per control area and for placing portions of the index adjacent to the data set.

VSAM fixes the number of control intervals for each control area in the data set. For a key-sequenced data set, the size of a control area is determined on the basis of the space-allocation request, user-specified or default index and data control-interval size, and available buffer space. For a data set that must reside in its own data space, however, the size of a control area is always one cylinder.

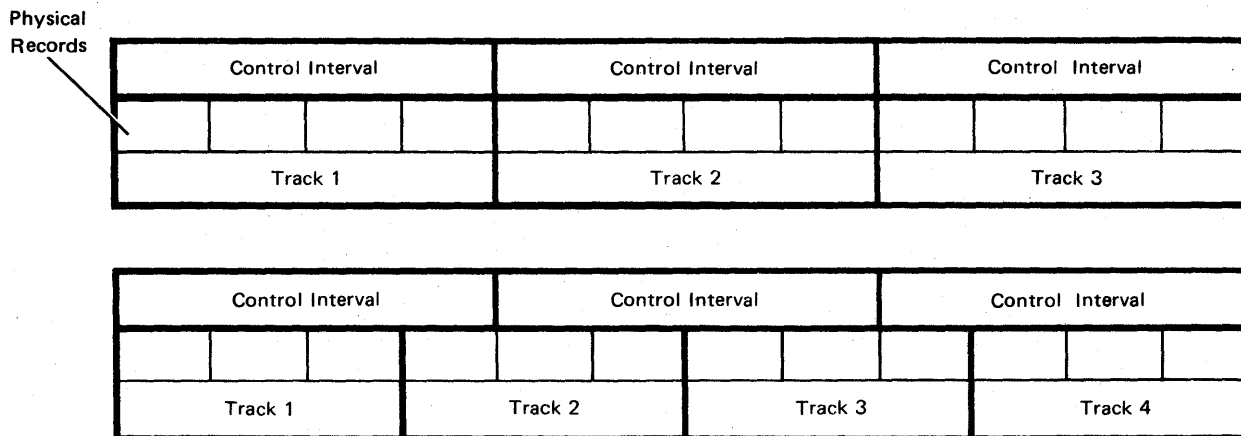


Figure 2. Control Intervals Are Independent of Physical Record Size

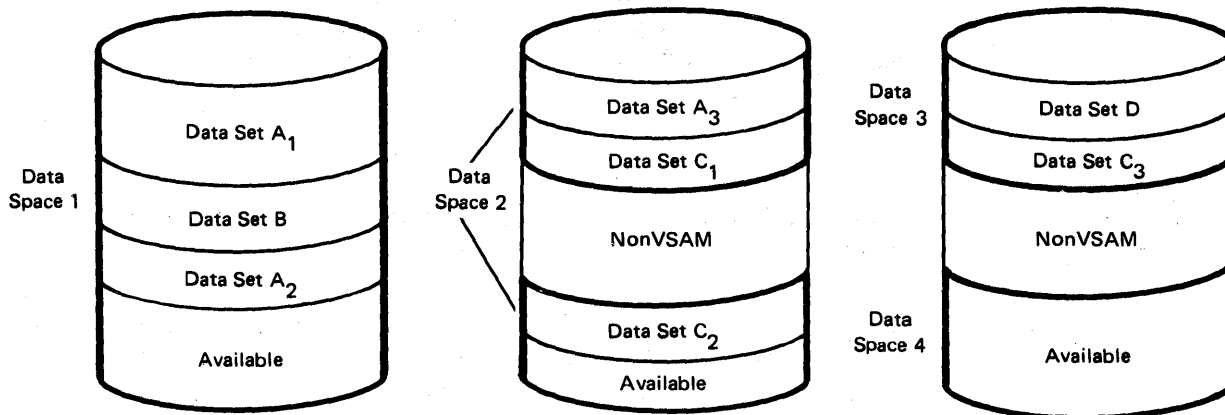


Figure 3. Relationship Among Storage Volumes, Data Spaces, and Data Sets

The Method of Storing a Record in a Control Interval

The records of a data set may be either fixed or variable in length: VSAM treats them the same. It puts control information at the back of a control interval to describe the data records stored in the control interval: the combination of a data record and its control information, though they are not physically adjacent, is called a *stored record*. (When adjacent records are the same length, they share control information.) Figure 4 shows how data records and control information are stored in a control interval. The data records are stored in the front of a control interval, and control information in the back.

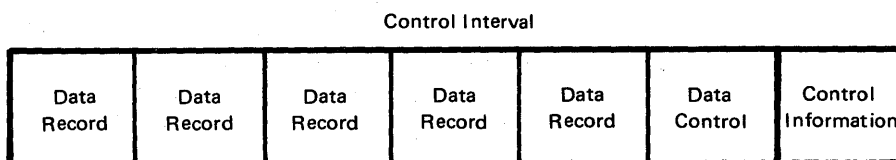


Figure 4. Placement of Data Records and Control Information in a Control Interval

Stored records do not span control intervals. When you define a data set, you must specify enough buffer space for the control intervals in the data set to be large enough for your largest stored record. The maximum control-interval size is 32,768 bytes.

A data record is addressed not by its location in terms of the physical attributes of the storage device (such as the number of tracks per cylinder), but by its displacement, in bytes, from the beginning of the data set, called its *RBA (relative byte address)*. The RBA does not depend on how many extents belong to the data set or on whether they are in different data spaces or on different volumes. For relative byte addressing, VSAM considers the control intervals in the data set to be contiguous, as though the data set were stored in virtual storage beginning at address 0.

Key-Sequenced and Entry-Sequenced Data Sets

The purpose of this section is to describe VSAM's two types of data sets in detail and to explain further how VSAM uses the control interval for data storage. Figure 5 contrasts the two types of data sets by listing the attributes of each.

Key-Sequenced Data Set	Entry-Sequenced Data Set
Records are in collating sequence by key field	Records are in the order in which they are entered
Access is by key through an index or by RBA	Access is by RBA
A record's RBA can change	A record's RBA cannot change
Distributed free space is used for inserting records and changing their length in place	Space at the end of the data set is used for adding records
Space given up by a deleted or shortened record is automatically reclaimed	A record cannot be deleted, but you can reuse its space for a record of the same length

Figure 5. Comparison of a Key-Sequenced and Entry-Sequenced Data Set

Key-Sequenced Data Sets

A key-sequenced data set is always defined with an index, which provides a directory that relates key values to the relative locations of the data records in a data set. The index and distributed free space used to insert a new record in key sequence are discussed in the paragraphs that follow.

An index relates key values to the relative locations of the data records. A key in the index is taken from a record's key field, whose size and position are the same for every record in the data set, and whose value cannot be altered. VSAM uses an index to locate a record for retrieval and to locate the collating position for insertion.

An index has one or more levels, each of which is a set of records that contain entries giving the location of the records in the next lower level. The index records in the lowest level are collectively called the *sequence set*; they give the location of control intervals. The records in all the higher levels are collectively called the *index set*; they give the location of index records. The highest level always has only a single record. The index of a data set with few enough control intervals for a single sequence-set record has only one level: the sequence set itself.

Figure 6 illustrates the levels of an index and shows the relationship between a sequence-set index record and a control area. The figure shows that The highest-level index record (A) controls the entire next level (records B through Z); each sequence-set index record controls a control area.

An entry in an index-set record consists of the highest key that an index record in the next lower level contains, paired with a pointer to that index record. An entry in a sequence-set record consists of the highest key in a control interval, paired with a pointer to that control interval. Not all data records have sequence-set entries, for there is only one entry for each control interval in the data set.

For direct access by key, VSAM follows *vertical pointers* from the highest level down to the sequence set to find a vertical pointer to data; for sequential

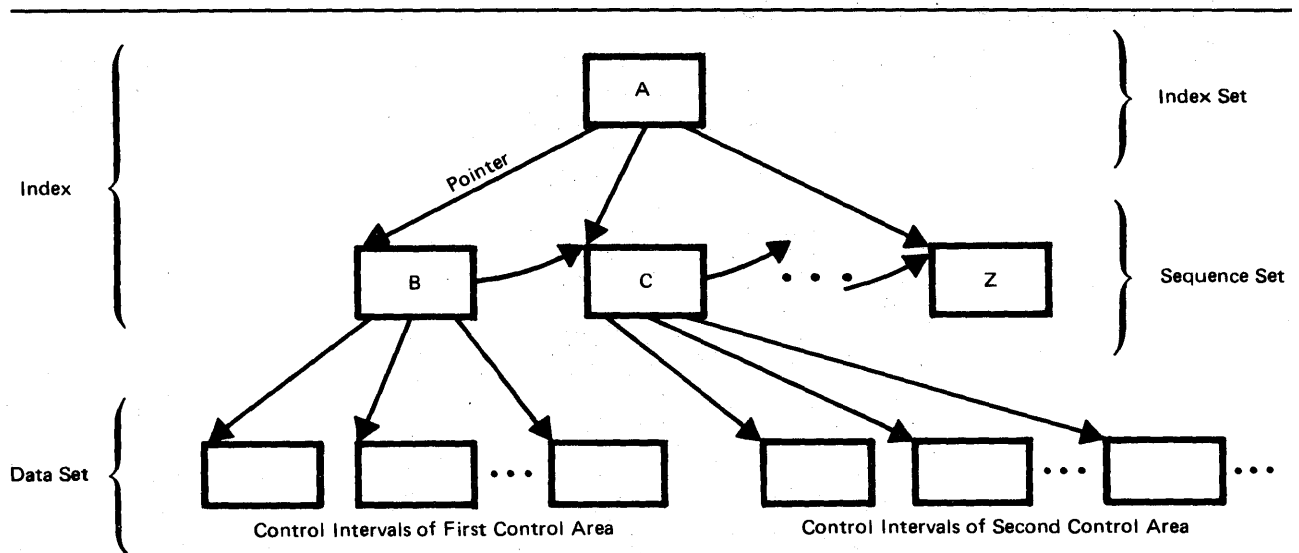


Figure 6. Relationship Among the Levels of an Index and a Data Set

access by key, VSAM refers only to the sequence set. It uses a *horizontal pointer* in a sequence-set record to get from that sequence-set record to the one containing the next key in collating sequence so it can find a vertical pointer to data. Figure 6 shows both vertical pointers and horizontal pointers.

VSAM increases the number of entries that an index record of a given size can hold by a method of *key compression*: it eliminates from the front and the back of a key those characters that aren't necessary to distinguish it from the adjacent keys. Compression helps achieve a physically smaller index by reducing the size of keys in index entries. For example, a two-level index, the size of whose records is 2048 bytes, with key field of 16 bytes, and the size of whose entries, including compressed key and pointer, is 8 bytes on the average, can control approximately 62,500 control intervals, each of which may contain numerous data records.

The number of control intervals in a control area equals the number of entries in a sequence-set index record. This equality has important uses in:

- Placing the sequence-set index record adjacent to the control area on a single cylinder (see "Sequence-Set Records Adjacent to the Data Set" in the chapter "Optimizing the Performance of VSAM")
- Distributing free space throughout a data set as a percent of free control intervals in each control area

When you define a key-sequenced data set, you can specify that free space is to be distributed throughout it in two ways: by leaving some space at the end of all the used control intervals and by leaving some control intervals completely empty. The amount of free space in a used control interval and the number of free control intervals in a control area are independent of each other. Figure 7 shows how free space might be set aside in each control area of a data set. The sequence-set record for a control area contains an entry for each free control interval, as well as for each of those that contain data.

Besides the space that you distribute when you create a key-sequenced data set, space that becomes available within a control interval when a record is shortened or deleted from the data set is automatically reclaimed by VSAM and can be used when a record is lengthened in place or inserted into the

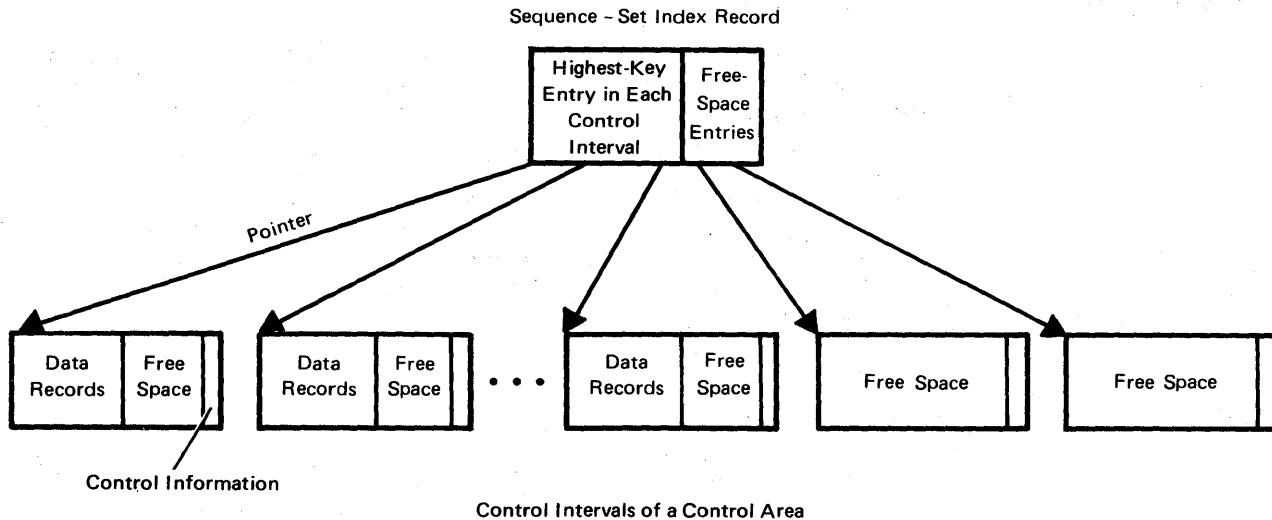


Figure 7. Distribution of Free Space in a Key-Sequenced Data Set

control interval. To reclaim space taken up by empty control intervals or control areas, however, you must reorganize the data set.

Reclaiming space and using distributed free space may cause RBAs of some records to change. As Figure 7 illustrates, free space within a used control interval is between the data in the front and the control information in the back. If a record is deleted or shortened, any succeeding records in the control interval are moved to the left and their RBAs are changed so that the space vacated can be combined with the free space already in the control interval. Conversely, an insertion or a lengthening causes any succeeding records in the control interval to be moved to the right into free space and their RBAs to be changed.

Entry-Sequenced Data Sets

No index is associated with an entry-sequenced data set. When a record is loaded or subsequently added, VSAM indicates its RBA to you. You must keep track of the RBAs of the records yourself to gain access to them directly. One way to keep track is to build your own index.

Sequential access with an entry-sequenced data set is similar to that of QSAM (queued sequential access method).

You can use direct access with an entry-sequenced data set in a way similar to BDAM (basic direct access method) by preformatting the data set with records of your choice (filled with blanks, for instance) and providing a routine that randomly associates an RBA with the key field of a record in the data set and thus distributes records throughout the data set.

To store a record initially, you convert its key field to an RBA, retrieve the preformatted record at that RBA, and store the new record back at that RBA. The routine must have a procedure for determining an alternate RBA when two or more keys are converted to the same RBA. To retrieve a record subsequently, you convert its key field to its RBA and determine the alternate RBA, if one is required.

How Are VSAM Data Sets Created?

This short discussion on creating data sets is intended merely to introduce the following description of data access and VSAM catalogs. See “How Is Access Method Services Used?” in the chapter “Communicating with VSAM” for a more detailed discussion of defining data sets and loading records into them.

To define a VSAM data set, you use Access Method Services to allocate storage space for it and catalog it in either the master catalog or a user catalog. You can load data records into a data set by having Access Method Services copy them from a sequential, an indexed sequential, or another VSAM data set, or you can load them with your own processing program.

In What Ways Can VSAM Data Sets Be Processed?

VSAM allows both sequential and direct access for each of its two types of data sets. Sequential access of a record depends on the position, with respect to the key or the address, of the previously processed record; direct access does not. With sequential access, records retrieved by key are in key sequence; records retrieved by address are in entry sequence. To retrieve records after initial positioning, you don't need to specify a key or an RBA. VSAM automatically retrieves or stores the next record in order—either next in key sequence or next in entry sequence, depending on whether you're processing by key or by address. With direct access, the retrieval or storage of a record is not dependent on the key or the address of any previously retrieved record. You must fully identify the record to be retrieved or stored by key or by address.

VSAM allows a processing program or its subtasks to process a data set with multiple concurrent sequential and/or direct requests, each requiring that VSAM keep track of a position in the data set, with a single opening of the data set. Access can be to the same part or to different parts of a data set. See “How Is Shared Data Protected?” in “Protecting Data with VSAM” for information about how VSAM provides for the protection of shared data.

For a key-sequenced data set, the primary form of access is keyed access, using the index; for an entry-sequenced data set, the only form of access is addressed access, using the RBA determined for a record when it was stored in the data set. You can also use addressed access to process a key-sequenced data set, but because previous keyed insertion and deletion may change the RBAs of records, you must keep track of each record's RBA by providing a routine to record RBA changes during processing. VSAM exits to the routine at the appropriate time.

When your processing program retrieves a record, VSAM reads into virtual storage the contents of the whole control interval in which it is stored (unless the contents have been read in previously). VSAM does not require the processing program to deblock records. VSAM indicates the length of the data record to your program and either places the record in your program's work area or gives your program the record's address in VSAM's I/O buffer. You need not concern yourself with any physical attributes of stored records.

VSAM provides programmers of utilities and systems with *control-interval access*. They retrieve and store the contents of a control interval, rather than a single data record, by specifying control-interval access in the macros. Access to control intervals is gained sequentially, or directly by RBA. They are responsible for maintaining the control information in the control interval;

detailed information on how to maintain control information in the control interval will be available when VSAM is released. The format of the information may change in future releases of VSAM.

If you must manage your own I/O buffers, you may. For additional information, see “RPL: Defining the Request Parameter List” in the chapter “Communicating with VSAM.” Detailed information on how to manage your own I/O buffers will be available when VSAM is released.

Keyed Access for Key-Sequenced Data Sets

Keyed access is only for a key-sequenced data set with an index. An entry-sequenced data set has no index, and thus cannot be processed by keyed access.

Keyed access provides for retrieval, update (including lengthening or shortening a record, as well as altering its contents), insertion, addition, and deletion. Each of these actions can be sequential, skip sequential, or direct.

Keyed Sequential Retrieval and Keyed Direct Retrieval

Keyed sequential access depends on where the previous macro request positioned VSAM with respect to the key sequence defined by the index. When the processing program opens the data set and its index for keyed access, VSAM is positioned at the first record in the data set in key sequence to begin keyed sequential processing. The POINT macro instruction positions VSAM for keyed sequential processing at the record whose key is specified. If the key specified is a leading portion of a key, or *generic key*, the record positioned to is the first of the records having the generic key. A subsequent GET macro retrieves the record VSAM is positioned at. The GET itself positions VSAM at the next record in key sequence.

With direct access, you may optionally specify for GET to position VSAM at the next record in key sequence: your program can then process the following records sequentially.

For retrieving records in sequence throughout a data set, the processing program can specify *skip sequential access*. When the program indicates the key of the next record to be retrieved, VSAM skips to its index entry by using horizontal pointers in the sequence set to get to the appropriate sequence set index record to scan its entries.

Direct retrieval does not depend on previous positioning; VSAM follows vertical pointers from the highest level of the index down to the sequence set to retrieve a record that is specified entirely by the present request. The record to be retrieved can be specified by:

- The exact key of the record
- An approximate key less than or equal to the key field of the record
- A leading portion of the key (generic key) of the record

Approximate specification can be used when the exact key is unknown. If a record actually has the key specified, VSAM retrieves it; otherwise it retrieves the record with the next greater key. Generic-key specification for direct access causes VSAM to retrieve the first record having that generic key. If all the records with the generic key are to be retrieved, the processing program should shift to sequential access to retrieve the rest of the records.

Keyed Sequential Deletion and Keyed Direct Deletion

An ERASE macro instruction following a GET for update deletes the record that the GET retrieved. To delete a record, you must previously have retrieved it for update.

Keyed Sequential Storage and Keyed Direct Storage

A PUT macro instruction stores a record. A PUT for update following a GET for update stores the record that the GET retrieved. To update a record, you must previously have retrieved it for update.

When VSAM detects that two or more records are to be inserted in sequence into a collating position in a data set, VSAM uses a technique called *mass sequential insertion* to buffer the records being inserted to save I/O operations. Using sequential instead of direct access to insert two or more records in sequence between two records in a data set enables you to take advantage of this technique.

To store records in sequence in collating positions throughout a data set, you can use sequential, skip sequential, or direct access. With sequential or skip sequential access, VSAM skips to the next collating position by scanning the sequence set of the index; with direct access, it finds the next collating position by searching the index from top to bottom.

The Use of Free Space for Processing a Key-Sequenced Data Set

VSAM uses free space for efficient insertion and lengthening of records in a key-sequenced data set and automatically combines the space that is given up by deletion or shortening of records with any free space already in the affected control interval.

The following discussion applies to insertion of a new record and to lengthening of an existing record. For simplicity, only insertion is referred to explicitly.

The simplest case is insertion of a record into a control interval that has enough free space to accommodate the record. Depending on the relationship of the key of the new record to the keys of existing records in the control interval, VSAM may move some of the existing records over to keep the records physically in key sequence within the control interval.

If the record to be inserted will not fit in the control interval, there is a *control-interval split*: VSAM moves stored records in the control interval to an empty control interval in the same control area, and inserts the new record in its proper key sequence. The number of records moved depends on the position of insertion of the new record and on the type of insertion. For sequential insertion, records are inserted in the control interval leaving any specified free space; when the next record to be inserted will not fit in the control interval, the records with keys greater than the key of the record to be inserted are moved to a new control interval. The new record is inserted in the old control interval. If there are no records in the control interval with a key greater than the new record, the new record is placed in a new control interval. an empty control interval. For direct insertion, approximately half of the records in a control interval are moved when a control-interval split is required.

Figure 8 illustrates a control-interval split and shows the resulting free space available in the two affected control intervals. Some of the records in the control interval that is too full for insertion are moved to a free control interval, and the new record is inserted into the control interval according to key sequence. Because the number of records in the first control interval is reduced, subsequent insertions revert to the simpler case, instead of becoming more complex.

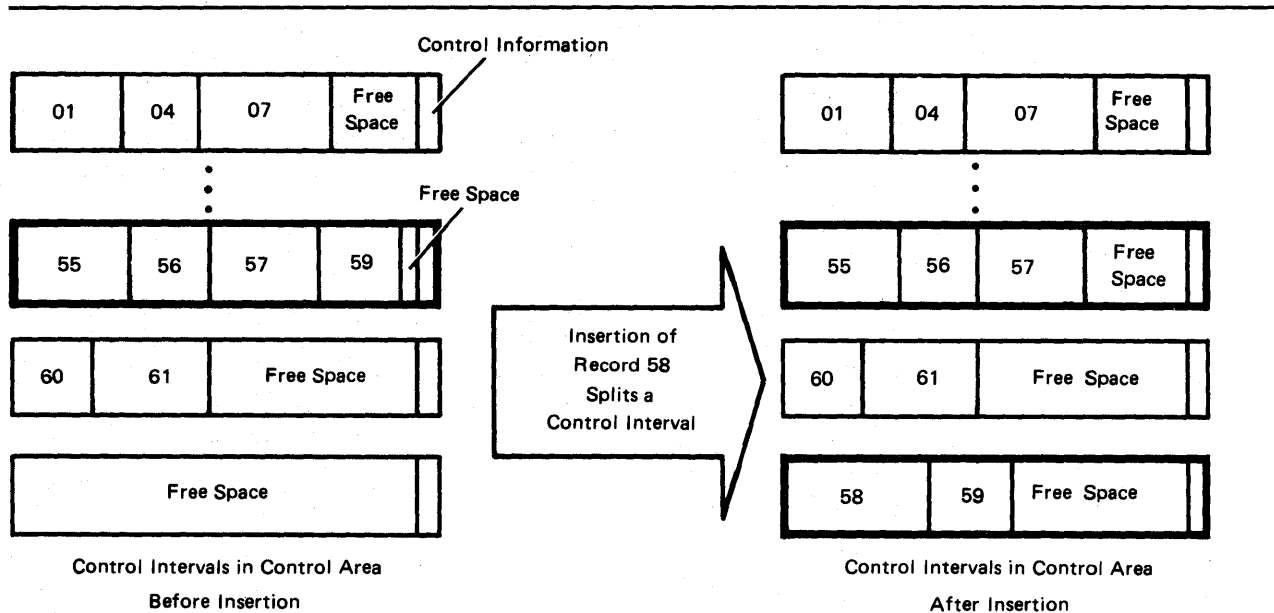


Figure 8. Splitting a Control Interval for Record Insertion

If the control intervals involved in a split are not adjacent, the entry sequence of data records is no longer the same as their key sequence. In Figure 8, the entry sequence of the records in the last three control intervals on the right is: 55, 56, 57, 60, 61, 58, 59. But the sequence-set index record reflects the key sequence, so that, for keyed sequential requests, the data records are retrieved in the order: 55, 56, 57, 58, 59, 60, 61.

Should there not be a free control interval in the control area, an insertion requiring a free control interval causes a *control-area split*: VSAM establishes a new control area, either by using space already allocated or by extending the data set, if the initially allocated space is full and you provided for extensions when you defined the data set. VSAM moves the contents of approximately half of the control intervals in the full control area to free control intervals in the new control area and inserts the new record into one of the two control areas, as its key dictates. Since about half of the control intervals of each of these control areas are now free, subsequent insertions won't require control-area splitting. Splitting should be an infrequent occurrence for data sets with sufficient distributed free space; splitting a control area does make it possible, however, to insert records into a key-sequenced data set without previously distributed free space.

Addressed Access for Both Types of Data Sets

Addressed access can be sequential or direct and can be used to gain access to key-sequenced and entry-sequenced data sets, but actions allowed for a key-sequenced data set are different from the actions allowed for an entry-sequenced data set. With a key-sequenced data set, you can use addressed access for retrieval, update of the contents of a record (except its key field), and deletion. With an entry-sequenced data set, you can use addressed access for retrieval, update of the contents of a record, and addition of a new record at the end of the data set.

The discussions of free space in a key-sequenced data set pointed out some of the ways keyed access may change RBAs. If you use addressed access to process a key-sequenced data set, you should consider the possibility that RBAs may have changed during previous keyed access.

Addressed Sequential Retrieval and Addressed Direct Retrieval

Positioning for addressed sequential retrieval is done by entry sequence rather than by key sequence. When a processing program opens a data set for addressed access, VSAM is positioned at the first record in the data set in entry sequence to begin addressed sequential processing. A POINT positions VSAM for sequential access beginning at the record whose RBA is indicated. A sequential GET causes VSAM to retrieve the data record at which it is positioned and positions VSAM at the next record in entry sequence.

With direct access, you may optionally specify for GET to position VSAM at the next record in entry sequence: your program can then process the following records sequentially.

Addressed sequential access retrieves records in entry sequence. If you use addressed sequential retrieval for a key-sequenced data set, you may not get records in their key sequence if there have been control-interval or control-area splits.

Addressed direct retrieval requires that the RBA of each individual record be specified, since previous positioning is not applicable. The address specified for a GET or a POINT must correspond with the beginning of a data record, otherwise the request is invalid.

Addressed Sequential Deletion and Addressed Direct Deletion

You can use the ERASE macro only with a key-sequenced data set to delete a record that you have previously retrieved for update.

With an entry-sequenced data set, you are responsible for marking inactive a record that you want to delete, using your own indication for inactivity. In other words, the record is inactive, not with respect to VSAM, but with respect to your application. You can reuse the space for a record marked inactive by retrieving the record for update and storing in its place a new record of the same length.

Addressed Sequential Storage and Addressed Direct Storage

VSAM does not insert new records into an entry-sequenced data set, but adds them at the end. With addressed access of a key-sequenced data set, VSAM does not insert or add new records.

A PUT macro instruction stores a record. A PUT for update following a GET for update stores the record that the GET retrieved. To update a record, you

must previously have retrieved it for update. You can update the contents of a record with addressed access, but you cannot alter the record's length. Neither can you alter the key field of a record in a key-sequenced data set.

With an entry-sequenced data set, if you change a record's length, you must store the record either at the end of the data set (as a new record) or in the place of an inactive record of the same length (as an update). You are responsible for marking inactive the old version of the record whose length you're changing.

What Are the Master Catalog and User Catalogs For?

A VSAM catalog is arranged as a key-sequenced data set with an index. A master catalog is required with VSAM, and any number of user catalogs are optional. Almost everything that is true of the master catalog is true of user catalogs, but user catalogs have special uses that we will discuss after we consider the general functions of a VSAM catalog.

A VSAM Catalog's Use in Data and Space Management

VSAM catalogs are a central information point for all VSAM data sets and the direct-access storage volumes containing them. The information describing a volume and the data sets on it is extensive enough to enable VSAM to allocate and deallocate data sets on the volumes without the volumes' being mounted on a device of the system.

VSAM catalogs provide VSAM with the information to allocate space for data sets, verify authorization to gain access to them, compile usage statistics on them, and relate RBAs to physical locations.

You must catalog all of your VSAM data sets and indexes in a VSAM catalog. That is, a data set's name and many more facts about it must be entered in the catalog when you define the data set. All VSAM data sets on a volume must be cataloged in the same VSAM catalog, either the master catalog or a user catalog. A data set has an entry in only one catalog.

Each VSAM catalog defines itself; that is, it contains an entry that describes itself. In addition to being defined in this way, the master catalog is pointed to by the system catalog, and user catalogs are pointed to by the master catalog.

When you execute a program to process a data set, the order in which catalogs are searched is:

1. Any user catalog or catalogs specified for the job step.
2. Any user catalog or catalogs specified for the job when none is specified for the job step.
3. The master catalog.
4. The system catalog.

Because VSAM catalogs are searched before the system catalog, it is recommended that you define both VSAM and nonVSAM data sets in a VSAM catalog to improve performance. Data sets of generation data groups cannot be defined in a VSAM catalog, however.

Note: Use caution in naming your data sets. Because the VSAM catalog is always searched first, it is possible to lose access to a data set cataloged in the system catalog if it has the same name as a data set cataloged in the VSAM catalog.

Figure 9 illustrates how data sets can be divided up for cataloging among the system catalog, the master catalog, and user catalogs.

You use Access Method Services to define both VSAM data sets and other data sets in a VSAM catalog. Access Method Services also allocates space for new VSAM data sets. With VSAM catalogs, you do not use JCL either to catalog data sets or to allocate space for them.

Information Contained in the Records of a Catalog

Besides data-set records, a VSAM catalog has records describing direct-access volumes in terms of the allocation of data spaces and the location of available space. VSAM can allocate and deallocate space on cataloged volumes that are not mounted. However, when allocating space to a data set, if there is not sufficient space available in the data space or data spaces on a volume, VSAM requires you to mount the volume so it can extend a data space.

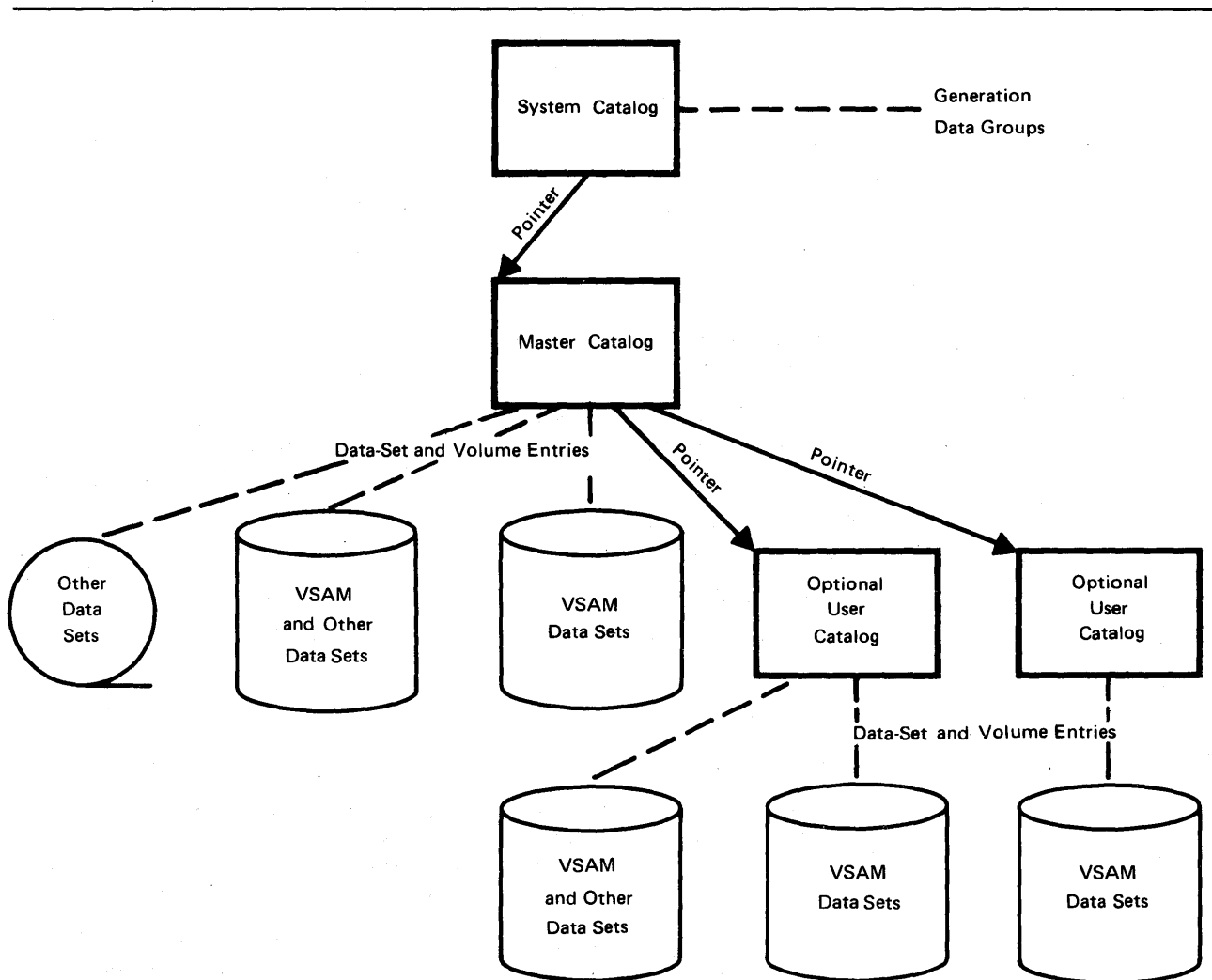


Figure 9. Cataloging VSAM and NonVSAM Data Sets in a VSAM Catalog

Information in a Data-Set Record

Data-set records provide the information required to make the connection between a data record's RBA and its physical location in terms of a storage volume's physical attributes. Besides the type of storage device and a list of volume serial numbers (which the system catalog contains for each data set cataloged in it), a VSAM catalog keeps other data-set information, including:

- A pointer to the location of each extent of the data set
- Statistics on the results of operations performed on the data set and its records, such as the number of insertions and deletions and the amount of free space remaining
- Attributes of the data set determined when it was defined, such as control-interval size, physical record size, number of control intervals in a control area, and, for a key-sequenced data set, location of the key field
- Password protection information
- An indication of the connection of a key-sequenced data set and its index
- Information used to determine whether either a key-sequenced data set or its index has been processed without the other
- Information about the volume(s) on which the data set is stored

Information in a Volume Record

Volume information in a VSAM catalog provides the information required to keep track of data spaces and free storage areas. An VSAM catalog contains this sort of volume information:

- The volume serial number and device characteristics
- The count of data spaces and data sets on a volume
- The location of data spaces on a volume
- The location of data sets within data spaces on a volume
- The location and size of free areas available for allocation to data sets
- An indication of the data spaces associated with a data set

The Special Uses of User Catalogs

User catalogs can improve catalog-processing performance and VSAM reliability and facilitate volume portability.

Improving Performance and Reliability

A large number of requests for information from a VSAM catalog may result in some of the requests being answered more slowly than they would be if several catalogs had parts of the information. You might have the master catalog primarily contain pointers to user catalogs, which would contain entries for most data sets, indexes, and volumes. By decentralizing data-set entries, you also reduce the time required to search a given catalog and minimize the effect of a catalog's being inoperative or unavailable.

Moving Volumes from One Operating System to Another

Because all VSAM data sets must be cataloged, moving a volume from one operating system to another requires that catalog information describing the volume and the data sets on it be moved along with the volume.

If you want to be able to move a volume or volumes from one OS/VS system to another, define a user catalog on one of the volumes and define the volumes and the VSAM data sets on them in the user catalog. You can then transport the volumes by demounting them and removing them from the first system, taking them to the second system, and remounting them. You use Access Method Services to disconnect the user catalog from the master catalog of the first system and to define a pointer to it in the master catalog of the second system. Any number of user catalogs can be used in this way.

You can move volumes between OS/VS and DOS/VS systems, but user catalogs are not used with VSAM on DOS/VS. You must treat the user catalog from an OS/VS system as the master catalog of a DOS/VS system.

You can also move individual data sets from one system to another by using Access Method Services. (See "Moving Data Sets from One Operating System to Another" in the chapter "Communicating with VSAM.") But the use of user catalogs for single volume portability is the most convenient way to achieve data-set portability.

COMMUNICATING WITH VSAM

This chapter introduces programmers to communicating with VSAM by using the commands of Access Method Services, the macros for connecting a processing program to a data set and gaining access to it, and the JCL parameters affected by VSAM. An application programmer doesn't need to know the format of control blocks, as he does with some other access methods: he just specifies the name of the action he wants.

How is Access Method Services Used?

Access Method Services is a multifunction service program that you use to define a VSAM data set and load records into it, convert a sequential or an indexed sequential data set to the VSAM format, list VSAM catalog records or records of a data set, copy a data set for reorganization, create a backup copy of a data set, recover from certain types of damage to a data set, and make a data set portable from one operating system to another.

Note: Access Method Services does not copy VSAM catalogs. For the procedure to be used to provide a backup copy of the catalog, see "Protecting the Catalog" in the chapter "Protecting Data with VSAM."

You tell Access Method Services what to do by giving a command and descriptive parameters through an input job stream or by calling it in a processing program and passing it a command statement. You can also execute Access Method Services from a TSO (Time Sharing Option) terminal, either by executing a program that calls it or by executing it directly and giving commands and parameters through an input data set, which can come from a TSO terminal. For more information about the use of TSO with VSAM, see "How Can the Time Sharing Option (TSO) Be Used with VSAM?" in the chapter "Preparing for VSAM."

A set of conditional statements (IF, ELSE, DO, END, SET) allow you to alter the sequence of execution of a series of commands by testing or resetting codes that Access Method Services sets to indicate the completion status of each command.

There are sets of Access Method Services commands for:

- Defining and deleting data sets and listing catalog records
- Copying and listing data sets
- Moving data sets from one operating system to another
- Recovering from damage to data

Defining and Deleting Data Sets and Listing Catalog Entries

You must use Access Method Services to define all VSAM data spaces, data sets, indexes, and catalogs. It makes entries for them in a VSAM catalog and allocates space for them. Four commands enable you to define data sets, alter the definitions, allocate and free auxiliary-storage space, and list catalog records: DEFINE, ALTER, DELETE, and LISTCAT.

DEFINE: Defining a Data Set and Allocating Space

To define a data space, entry-sequenced data set, key-sequenced data set and its index, or catalog, you specify the DEFINE command, the object to be defined, and, optionally, the catalog that is to contain an entry defining it. You define the relationship between a key-sequenced data set and its index, as well as define the two themselves, with a single DEFINE command. You also use DEFINE to catalog data sets of other access methods in a VSAM catalog.

There are parameters for specifying initial auxiliary-storage allocation, amount of space for extensions, erasure of data in a deleted data set, passwords and other authorization information, size and other attributes of data records, minimum amount of virtual-storage space for I/O buffers, percents of free space in control intervals and control areas of a key-sequenced data set and other performance options, retention period, identification of the owner of the data set defined in the entry, how a data set can be shared across regions or systems, data-set preformatting options, and whether write operations are to be verified.

You specify the amount of auxiliary-storage space for the object you are defining as the number of data records that the object is to contain or as a number of physical units, such as tracks or cylinders. Specifying the number of records, independent of type of storage device, leaves the calculation of the number of physical units of space up to VSAM. It calculates the size of the control interval and control area to be used. You may specify the control-interval size, and VSAM will use it so long as the size falls within the acceptable limits that VSAM calculates.

When you define a key-sequenced data set, you may specify that its space is to be allocated on volumes according to ranges of key values. The space for each range is extended separately when additional space is required.

For convenience, you may specify an existing catalog record as a model for a new entry, so long as they are of the same type (entry-sequenced data set, key-sequenced data set and its index, or user catalog). The information in the model will be used in the new entry unless you override it.

ALTER: Modifying a Catalog Entry

Many of the attributes that you define, either explicitly or by default, when you create a catalog record may be modified subsequently by way of the ALTER command, most of whose parameters are the same as the DEFINE parameters. You can change the name of a data set, the indication of whether to erase the data in a deleted data set, passwords and other authorization information, minimum amount of virtual-storage space for I/O buffers (which you may increase, but not decrease), percents of free space in new control intervals and control areas of a key-sequenced data set, retention period, name of the owner of a data set, the indication of how to share a data set, and the indication of whether to verify write operations.

Certain attributes of the data set, such as control-interval size and placement of the index in auxiliary storage relative to a key-sequenced data set, cannot be modified. Changing these attributes amounts to a reorganization of the data set and requires that you define a new data set and copy the old data set into it.

DELETE: Removing a Catalog Entry and Freeing Space

The DELETE command enables you to remove the entry for any previously defined object and, in effect, cause it to cease to exist. The space is freed for use by new objects and, if the erase option is specified in the entry, overwritten with binary 0s.

You must use Access Method Services to delete data spaces, data sets, indexes, and catalogs: you cannot delete them by way of the JCL disposition parameter or operating-system utilities.

LISTCAT: Listing Catalog Entries

The LISTCAT command enables you to list individual entries, all entries of a particular type, or all entries of a given catalog. You see the entire entry, except that passwords in an entry are not listed unless you specify the master password for the data set defined by the entry or the master password for the catalog itself.

Copying and Listing Data Sets

The REPRO and PRINT commands enable you to copy and list sequential, indexed sequential, and VSAM data sets.

REPRO: Converting and Reorganizing Data Sets

The REPRO command instructs Access Method Services to get records from a sequential, indexed sequential, or VSAM data set (other than a VSAM catalog) and put them into a sequential or VSAM data set. You may use it to convert an indexed sequential data set to a key-sequenced data set with an index. First, define a new key-sequenced data set and its index. Then copy the indexed sequential data set into the key-sequenced data set. Access Method Services converts data records to the VSAM format and builds an index.

You can reorganize an old data set by copying it into a newly defined data set of the same type. With key-sequenced data sets, you can optionally specify different percents of distributed free space and different performance options for the new data set. Copying the old key-sequenced data set into the new one redistributes free space, makes the entry sequence of the data records the same as their key sequence, and builds a new index.

The data set into which records are copied may either be newly allocated (by way of the DEFINE command) or contain records already. Records copied into a key-sequenced data set are merged with any existing records; records are added at the end of an entry-sequenced data set. You may specify a range of records to be copied by number of records, by key in an indexed sequential or a key-sequenced data set, or by address in either type of VSAM data set.

PRINT: Listing Data Records

The PRINT command instructs Access Method Services to list some or all of the records of a sequential, indexed sequential, or VSAM data set in one of three formats: each byte as 2 hexadecimal digits, each byte as a single character, or a combination of these two, side-by-side. You may specify a range of records for listing as you do for copying.

Moving Data Sets from One Operating System to Another

We discussed volume portability between OS/VS systems and between OS/VS and DOS/VS systems in “The Special Uses of User Catalogs” in the chapter “Getting to Know What VSAM Is and Does.” The EXPORT and IMPORT commands allow you to transport individual data sets between OS/VS systems or between OS/VS and DOS/VS systems, except that user catalogs are not supported by DOS/VS. Figure 10 compares volume and data-set portability. Data portability is achieved by moving volumes or by moving individual data sets.

EXPORT: Extracting Catalog Information and Making a Data Set Portable

The EXPORT command instructs Access Method Services to copy an entry-sequenced data set or a key-sequenced data set and its index (other than a VSAM catalog) in the format of a sequential data set onto a storage volume to be transported to another operating system. The transporting volume may be magnetic tape or disk. Access Method Services also extracts information from the catalog entry that defines the object to be transported and copies it onto the transporting volume. The information is used to define the object automatically in a VSAM catalog in the other operating system.

Exportation is either permanent or temporary. In permanent exportation, Access Method Services deletes the catalog record and frees the storage space; in temporary exportation of an object, both the sending and the receiving operating systems have a copy of it, and you may specify that one or both of the copies are not to be modified. A copy so protected can only be read. You may free the copy for full access with the ALTER command.

You use EXPORT to disconnect a user catalog from a master catalog when you are moving the user catalog to another system. The user catalog is not copied, but remains on its original volume in its original form.

IMPORT: Loading a Portable Data Set and Its Catalog Information

The IMPORT command instructs Access Method Services to define the entry-sequenced data set or the key-sequenced data set and its index on the transporting volume in the catalog that you specify, using the catalog information extracted in exportation. The object itself is stored in its VSAM format in a data space that is defined in the specified catalog.

You use IMPORT to define a pointer to a user catalog in the master catalog. The user catalog is not copied, but remains on its original volume in its original form.

You can use the EXPORT and IMPORT commands to prepare a backup copy of an entry-sequenced data set and its catalog record or a key-sequenced data set, its index, and their catalog records and to load the backup copy if it is needed. When you import a backup copy, the catalog record is regenerated.

Recovering from Damage to Data

With the Access Method Services VERIFY command, you can enhance the integrity of data when the data set was not closed. The VERIFY command causes end-of-file information in the catalog to be updated, which ensures that your data will not be overwritten inadvertently at a later date.

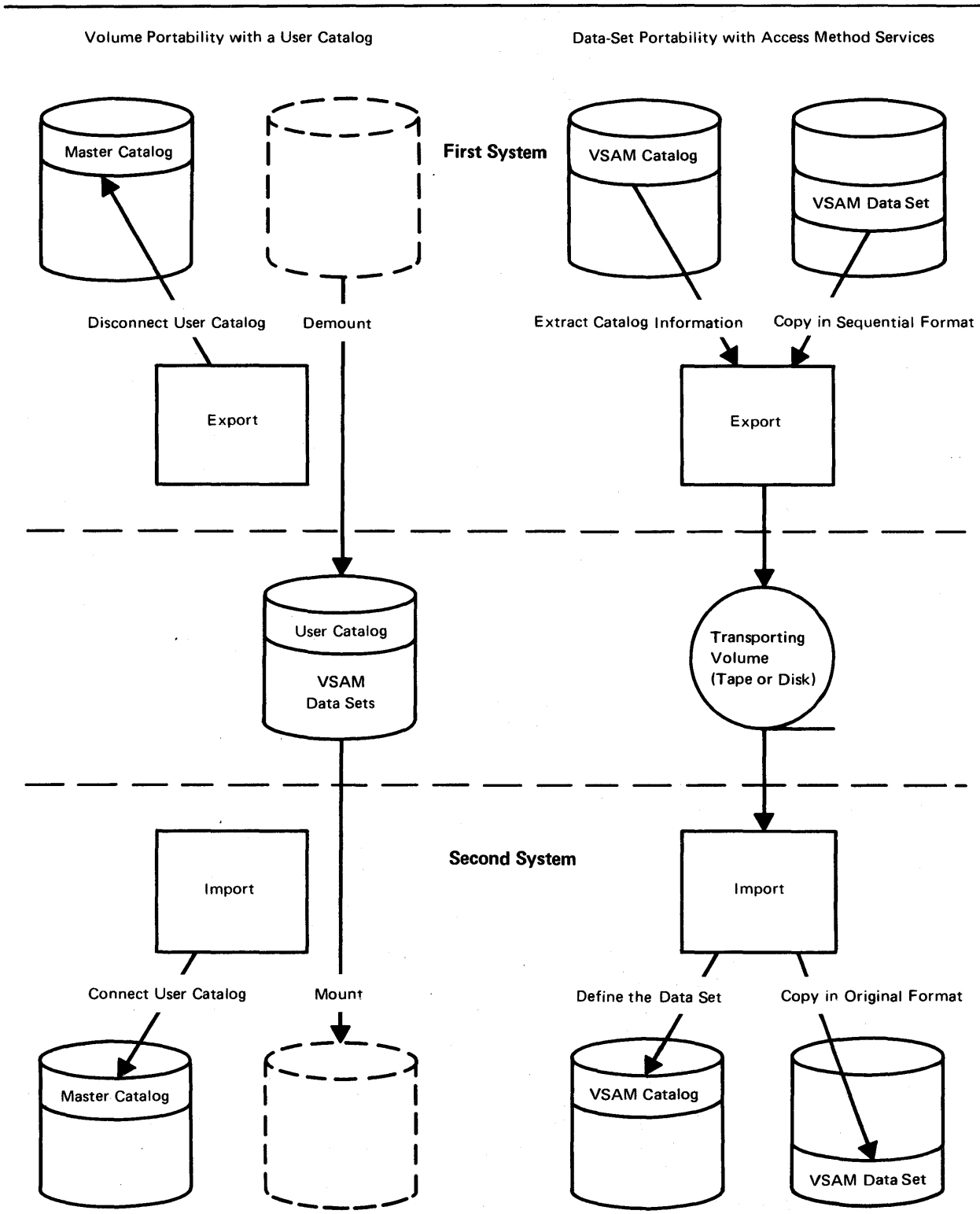


Figure 10. Comparison of Volume Portability and Data-Set Portability

VERIFY: Testing and Reestablishing a Data Set's Integrity

The end of a data set is indicated by an end-of-file indicator at the end of the data set and by information in the data set's catalog record. The end may be improperly indicated in the catalog if an error prevented VSAM from closing the data set. You can instruct Access Method Services to close the data set. It modifies the catalog information, if necessary, to correspond with the data set.

What Are the Macros for Processing a VSAM Data Set?

You code the VSAM macros in a processing program to gain access to your data. There are macros for:

- Connecting and disconnecting a processing program and a data set. These prepare a bridge for VSAM between the program and the data.
- Specifying parameters that relate the program and the data. These identify the data set and describe the kind of processing to be done.
- Manipulating the information relating the program and the data. These are used to specify changes in processing.
- Requesting access to a data set. These initiate the transfer of data between auxiliary and virtual storage.

Connecting and Disconnecting a Processing Program and a Data Set

You use the OPEN macro to connect a processing program to a data set, so VSAM can satisfy the program's requests for data; you use CLOSE to complete processing and free resources that were obtained by the Open routine.

OPEN: Connecting a Processing Program to a Data Set

The Open routine verifies that a processing program has the authority to process a data set, by calling a VSAM authorization routine and your own routine, if you have indicated one in the data set's catalog record.

Open constructs VSAM control blocks and loads into virtual storage the routines required for the processing that you indicate in the ACB macro (described after CLOSE). By examining the DD statement indicated by the ACB macro and the volume information in the catalog, Open calls for the necessary volumes to be mounted and checks whether each volume matches its catalog information. If you are opening a key-sequenced data set and its index, Open checks whether the data set has been updated separately from its index; if the data set and its index have been updated separately, the open is not allowed unless the control or master password is supplied.

CLOSE: Disconnecting a Processing Program from a Data Set

The Close routine completes any operations that are outstanding when a processing program issues a CLOSE macro for a data set. For instance, VSAM buffers index records and data records, so the contents of a control interval may need to be stored or an index record updated and stored.

Close updates the catalog for any changes in the attributes of a data set. The addition of records to a data set may cause its end-of-file indicator to change, in which case Close updates the end-of-file indicator in the catalog. These end-of-file indicators help ensure that the entire data set is accessible. If an

error prevents VSAM from updating the indicators, the data set is flagged as not properly closed. When a processing program subsequently issues an OPEN macro, it is given an error code indicating the failure. For more information on correcting this condition, see the discussion of the Access Method Services VERIFY command and “Method of Indicating the End of a Data Set” in the chapter “Protecting Data with VSAM.”

Close restores control blocks to the status that they had before the data set was opened, deletes from virtual storage the routines that Open loaded, and frees the virtual-storage space that Open used to construct VSAM control blocks.

You can issue a CLOSE macro (TYPE=T) to update the catalog. Processing may continue without reopening the data set.

Specifying Parameters That Relate the Program and the Data

To open a data set for processing, you must identify the data set and the types of processing to be done. You use the macros ACB, EXLST, and RPL to specify a data set you want to process, the types of access you want to use, the addresses of your own exit routines, and the specific options for a particular request. The GENCB macro can be used in place of the ACB, EXLST, or RPL macro to generate processing specifications during the execution of a processing program, rather than during assembly or compilation of the program.

ACB: Defining the Access-Method Control Block

You use the ACB macro to define a control block for each entry-sequenced data set or key-sequenced data set and its index that your processing program will gain access to. You give the name of the JCL DD statement that describes the entry-sequenced data set or the key-sequenced data set and its index, so the Open routine can connect the program to the data. If you use more than one ACB with the same DD statement, VSAM uses the same set of I/O buffers for all requests to the specified data set.

The other information that you specify enables Open to prepare for the kind of processing to be done by your program:

- The address of a list of exit-routine addresses that you supply. You use the EXLST macro, described next, to construct the list.
- For processing concurrent requests, the number of requests that are defined for processing the data set. (See the discussion of the RPL macro following EXLST.)
- The size of the virtual-storage space for I/O buffers and the number of I/O buffers that you are supplying for VSAM to process data and index records. A minimum of two buffers is required for data control intervals for a single request for an entry-sequenced data set. A minimum of three buffers is required for a key-sequenced data set, two for data control intervals and one for index records. For concurrent requests that require VSAM to keep track of multiple positions in a data set, each additional request requires a minimum of one buffer for control intervals and one buffer for index records. For example, three concurrent requests requires a minimum of four buffers for control intervals and three buffers for index records.
- The password that is required for the type of processing desired.

- The processing options to be used: keyed, addressed, or control interval, or a combination; sequential, direct, or skip sequential access, or a combination; retrieval, storage, or update (including deletion), or a combination.

EXLST: Defining the Exit List

You use the EXLST macro to specify the addresses of optional exit routines that you may supply for analyzing physical and logical errors, end-of-data-set processing, noting RBA changes, and writing a journal. Any number of ACB macros in a program may indicate the same exit list for the same exit routines to do all the special processing for them, or they may indicate different exit lists.

You can use exit routines for:

Analyzing physical errors. When VSAM encounters an error in an I/O operation that the operating system's error routine cannot correct, the error routine formats a message for your physical-error analysis routine to act on.

Analyzing logical errors. Errors not directly associated with an I/O operation, such as an invalid request, cause VSAM to exit to your logical-error analysis routine.

End-of-data-set processing. When your program requests a record beyond the last record in the data set, your end-of-data-set routine is given control. The end of the data set is beyond either the highest-addressed or the highest-keyed record, depending on whether your program is using addressed or keyed access.

Writing a journal. To journalize the transactions against a data set, you may specify a journal routine, which VSAM exits to before moving your data to the control-interval buffer. To process a key-sequenced data set by way of addressed access, you need to know whether any RBAs changed during keyed processing. When you're processing by key, VSAM exits to your routine for noting RBA changes before transmitting to auxiliary storage the contents of a control interval in which there is an RBA change.

RPL: Defining the Request Parameter List

The RPL macro defines the request parameter list, or the list of parameters required for a particular request for access. It identifies the data set to which the request is directed by naming the ACB macro that defines the data set.

You can use a single RPL macro to define parameters that apply to all of the requests (GET, PUT, POINT, and ERASE, described under "Requesting Access to a Data Set") for access to a data set. You use the MODCB macro (described following GENCB) to modify some of the parameters to change the type of processing. For example, you can change from direct to sequential or from update to nonupdate processing.

For concurrent requests that require that VSAM keep track of more than one position in a data set, you may use any number of RPL macros to specify requests that your processing program or its subtasks can issue asynchronously to gain access to the same data set concurrently. The requests can be sequential or direct or both, and they can be for records in the same or different parts of the data set.

You need specify only the RPL parameters appropriate to a given request, as follows:

Address of the next request parameter list in a chain. You can chain request parameter lists together to define a series of actions for a single GET, PUT, or ERASE. For example, each request parameter list in the chain could contain a unique search argument and point to a unique work area. A single GET macro would retrieve a record for each request parameter list in the chain. A chain of request parameter lists is processed as a single request (chaining request parameter lists is not the same as processing concurrent requests that require VSAM to keep track of multiple positions in a data set.)

Processing options for a request. A request is to gain access to a data record or a control interval. Access may be gained by address (RBA) or by key.

Addressed access may be sequential or direct; keyed access may be sequential, skip sequential, or direct. Access may be for updating or not updating. A nonupdate direct request to retrieve a record can optionally cause positioning at the following record for subsequent sequential access. The characteristics that may be specified are summarized, as follows:

- A request (including a request defined by a chain of request parameter lists) is either synchronous, so that VSAM does not give control back to your program until the request is completed, or asynchronous, so that your program may continue to process or issue other requests while the request is active and later use the CHECK macro to suspend processing until the request has been completed.
- For a keyed request, you specify either a generic key or a full key to which the key field of the record is to be matched. A generic search argument is matched for a less-than-or-equal comparison to the key field, and a full argument is matched for either an equal or a less-than-or-equal comparison to the key field.
- For retrieval, a request is either for a data record to be placed in a work area in the processing program or for the address of the record within VSAM's I/O buffer to be passed to the processing program. For all other requests (requests that involve updating or inserting) the work area contains the data record.
- For a request to gain direct access to a control interval, you specify an RBA, which must be on a control-interval boundary. With control-interval access, you are responsible for maintaining the control information in the control interval. If VSAM's buffers are used, VSAM allows control-interval and stored-record operations to be intermixed and provides exclusive control (that is, does not allow any other task to gain access), as required. If you provide your own buffers, intermixing is not allowed, and you must assume responsibility for exclusive control.

Address and size of the work area to contain a data record. You must provide a work area. It contains a data record or the address of the record within VSAM's I/O buffer. Having a work area that is too small is considered a logical error.

Length of the data record being processed. For storage, your processing program indicates the length to VSAM; for retrieval, VSAM indicates it to your program.

Length of the key. This parameter is required only for processing by generic key. For ordinary keyed access, the full key length is available to the Open routine from the catalog.

Address of the area containing the search argument. The search argument is either a key value or an RBA.

Address and length of an area for error messages from VSAM. Your routine for analyzing physical errors receives messages in this area.

GENCB: Generating Control Blocks and Lists

You use the GENCB macro in place of an ACB, EXLST, or RPL macro to generate an access-method control block, exit list, or request parameter list during the execution of your processing program, rather than producing it with the corresponding macro. You code GENCB the same as the other macros, but it enables you to generate one or more copies of a control block or list, and GENCB provides greater flexibility in the specification of parameter values.

Manipulating the Information Relating the Program and the Data

The macros MODCB, SHOWCB, and TESTCB are for modifying, displaying, and testing the contents of an access-method control block, exit list, or request parameter list.

MODCB: Modifying the Contents of Control Blocks and Lists

You use the MODCB macro to specify a new value for fields in an access-method control block, exit list, or request parameter list in the same way you defined them originally. For example, to use a single request parameter list to directly retrieve the first record having a certain generic key and then to sequentially retrieve the rest of the records having that generic key, you would use MODCB to alter the request parameter list to change from direct to sequential access.

SHOWCB: Displaying Fields of Control Blocks and Lists

SHOWCB allows you to examine the contents of fields in an access-method control block, exit list, or request parameter list. VSAM gives the contents to you in an area you provide and in the order you specify the fields. You may display the contents of fields additional to those that you define in the macros. For example, when a data set is open, you can display various counts, such as number of control-interval splits, number of deleted records, and number of index levels.

TESTCB: Testing the Contents of Control Blocks and Lists

The TESTCB macro enables you to test the contents of a field or combination of fields in an access-method control block, exit list, or request parameter list for a particular value and alter the sequence of your processing steps as a result of the test.

Requesting Access To a Data Set

All of the preceding macros are for preparing to process a data set. The request macros, GET, PUT, POINT, and ERASE, initiate an access to data. Each of these macros is associated with a request parameter list (or chain of request parameter lists) that fully defines the request: the only parameter that is specified with a request macro is the identity of the request parameter list.

The CHECK macro synchronizes a request initiated by a macro in the asynchronous form. In asynchronous processing, VSAM gives control back to

your program before completion of the request. You use CHECK to suspend processing, if necessary, until the request has been completed and to schedule any routines to handle unusual conditions. You use the ENDREQ macro to terminate a request that is not required to be completed or to free VSAM from keeping track of a position in a data set.

The options for using GET, PUT, POINT, and ERASE are outlined in the discussion of the RPL macro, and the use of each macro is discussed in the section “In What Ways Can VSAM Data Sets Be Processed?” in the chapter “Getting to Know What VSAM Is and Does.”

How is JCL Used?

VSAM uses a minimum number of JCL parameters. It has two optional DD statements, JOBCAT and STEPCAT, for specifying catalogs and an optional JCL DD parameter, AMP, for overriding parameters specified by a processing program.

Defining a VSAM Data Set

When you define a data set, no DD statement is required if Access Method Services can allocate space for the data set from an existing data space. If a data space must be created or extended to allocate space for the data set that you're defining, you need a DD statement for OS/VS job management to provide device allocation: you specify storage unit, volume, and a disposition of OLD. You never specify space parameters (SPACE, SPLIT, SUBALLOC) or a disposition of NEW, DELETE, CATLG, or UNCATLG, since you use Access Method Services to define and delete all VSAM data sets.

Processing a VSAM Data Set

The catalog contains most of the information required by VSAM to process a data set, so VSAM requires minimal information from JCL. Data-set name and disposition are sufficient to describe the data set. A key-sequenced data set and its index are defined by a single DD statement.

To limit a data set to access by a single job step, use a disposition of OLD. Use a disposition of SHR in the JCL of separate jobs to enable two or more job steps to share a data set, provided the data set's definition in the catalog specifies that sharing is permitted.

Specifying VSAM Catalogs

The master catalog is always available, without JCL specification. You make user catalogs available by describing them in DD statements with special names for a job or a job step: JOBCAT and STEPCAT. You describe a catalog sufficiently by giving its data-set name and a disposition of SHR. A user catalog may be either a STEPCAT or JOBCAT catalog; if both STEPCAT and JOBCAT user catalogs are specified, the STEPCAT catalog is available for the step for which it is specified, and the JOBCAT catalog is available for all steps for which no STEPCAT was specified. VSAM uses a data set's name as a search argument to search a catalog.

Using Other JCL Parameters

Some JCL parameters are ignored, are invalid, or bring about the wrong results if used with VSAM, and VSAM has a special JCL DD parameter, AMP.

JCL Parameters Not Used with VSAM

VSAM ignores parameters for defining tape data sets: data-set sequence number, NSL, NL, BLP, and AL. You may not use the parameters for a sequential data set, DATA, SYSOUT, and *, for specifying a VSAM data set. These DD names are invalid for VSAM data sets: JOBLIB, STEPLIB, SYSABEND, SYSUDUMP, and SYSCHK.

These DD parameters are also invalid: UCS, QNAME, DYNAM, TERM, and the forms of DSNAME for ISAM, PAM (partitioned access method), and generation data groups. VSAM does not use temporary data sets or concatenated data sets.

VSAM's Special DD Parameter: AMP

The VSAM DD parameter, AMP, has subparameters for specifying attributes that you can also specify by way of the ACB or the EXLST macros: size of virtual-storage space for I/O buffers, number of I/O buffers for data and index records, number of concurrent requests to be processed, and name of an exit routine for analyzing physical errors. AMP values override any values specified by way of the macros.

To mount only some of the volumes on which a VSAM data set is stored, you must specify the DD parameters VOLUME and UNIT. Specifying these parameters prevents a reference to the catalog and requires you to use another AMP subparameter to specify that the data set is a VSAM data set.

Another subparameter is used for specifying checkpoint/restart options. They are described in "How Are Programs Restarted Following a Failure?" in the chapter "Protecting Data with VSAM."

PREPARING FOR VSAM

This chapter indicates, for all prospective users of VSAM, the programming languages in which you can write programs to use VSAM, and the use of TSO (Time Sharing Option) and SMF (System Management Facilities) with VSAM.

The topic "How Can Existing Programs That Use ISAM Be Used with VSAM?" is for users of ISAM and may be ignored by other readers. It contains detailed information for programmers to decide whether existing programs that use ISAM can use the ISAM interface to process new key-sequenced data sets with indexes or key-sequenced data sets with indexes into which indexed sequential data sets have been converted.

What Programming Languages Can VSAM Be Used With?

You can use the OS/VS assembler programming language to code all of the macros of VSAM.

You can also code programs in PL/I and COBOL, using ISAM, to process VSAM data sets by way of the ISAM interface.

How Can the Time Sharing Option (TSO) Be Used with VSAM?

TSO is an optional subsystem of OS/VS2 that provides conversational time sharing from remote terminals. As a TSO user, you can:

- Execute a program using VSAM or ISAM macros to process a VSAM data set
- Execute Access Method Services (commands must be entered through an input data set, which can come from a TSO terminal, to Access Method Services)
- Execute a program to call Access Method Services
- Dynamically allocate a VSAM data set during the execution of a job step, provided a single volume contains the data set and is already mounted
- Allocate a VSAM data set by way of a LOGON procedure

VSAM data sets must be cataloged in the master catalog or in a user catalog. The master catalog is allocated when the system is initialized; you can allocate a user catalog by way of the LOGON procedure.

For details about writing and executing programs and allocating data sets with TSO, see *OS/VS2 TSO Terminal User's Guide*, GC28-0645, and *OS/VS2 TSO Command Language Reference*, GC28-0646.

How Can System Management Facilities (SMF) Be Used with VSAM?

SMF is an optional program of OS/VS that provides the means for gathering and recording information that can be used to evaluate system usage. VSAM supplies volume and data-set information to SMF. It also uses SMF to record changes to VSAM catalogs. See “Protecting the Catalog” in “Protecting Data with VSAM” for information on how SMF records can be used to rebuild a catalog from an earlier backup copy.

For further details about the facilities of SMF and how to use it, see *OS/VS System Management Facilities (SMF)*, GC35-0004.

How Can Existing Programs That Use ISAM Be Used with VSAM?

This section is intended for users of ISAM who are converting to VSAM. VSAM’s ISAM interface minimizes your conversion costs and scheduling problems by permitting programs coded to use ISAM to process VSAM data sets. To use the interface, you must convert indexed sequential data sets to VSAM data sets (for which you can use Access Method Services), convert ISAM JCL to VSAM JCL, and ensure that your existing ISAM programs meet the restrictions for using the interface.

Comparison of VSAM and ISAM

In most cases, if you use the performance options described in the chapter “Optimizing the Performance of VSAM,” you can get better performance with VSAM while achieving essentially the same results that you can achieve with ISAM; you can also achieve results that you can’t achieve with ISAM. The use of your existing ISAM processing programs to process key-sequenced data sets depends upon the extent to which VSAM and ISAM are similar in what they do, as well as upon the limitations of the ISAM interface itself. This subsection describes the similarities and differences between VSAM and ISAM in the areas that you are familiar with from using ISAM and indicates the functions of VSAM that have no counterpart in ISAM.

Comparison of VSAM and ISAM in Common Areas

A number of things that ISAM does are done differently or not at all by VSAM, even though the same practical results are achieved. The areas in which VSAM and ISAM differ are:

- Index structure
- Relation of index to data
- Deleting records
- Defining and loading a data set

These differences are described in the paragraphs that follow.

Index structure. Both a VSAM key-sequenced data set and an indexed sequential data set have an index that consists of levels, with a higher level controlling a lower level. In ISAM, either all or none of the index records of a higher level are kept in virtual storage. VSAM keeps individual index records in virtual storage, the number depending on the amount of buffer space you provide. It optimizes the use of the space by keeping those records it judges to be most useful at a particular time.

Relation of index to data. The relation of a VSAM index to the auxiliary-storage space whose records it controls is quite different from the corresponding relation for ISAM, with regard to overflow areas for record insertion. ISAM keeps a two-part index entry for each primary track that a data set is stored on. The first part of the entry indicates the highest-keyed record on the primary track. The second part indicates the highest-keyed record from that primary track that is in the overflow area for all the primary tracks on the cylinder and gives the physical location in the overflow area of the lowest-keyed record from that primary track. All the records in the overflow area from a primary track are chained together, from the lowest-keyed to the highest-keyed, by pointers that ISAM follows to locate an overflow record subsequently. Overflow records are unblocked, even if primary records are blocked. VSAM does not distinguish between primary and overflow areas. A control interval, whether used or free, has an entry in the sequence set, and after records are stored in a free control interval, it is processed exactly the same as other used control intervals. Data records are blocked in all control intervals and addressed, without chaining, by way of an index entry that contains the key (in compressed form) of the highest-keyed record in a control interval.

Deleting records. With ISAM, you mark records you want to delete, either for you to erase subsequently or for ISAM to drop, should they be moved into the overflow area; VSAM automatically reclaims the space in a key-sequenced data set and combines it with any existing free space in the affected control interval. Because of its use of distributed free space for insertions and deletions, VSAM requires less data-set reorganization than ISAM does. The ISAM interface allows you the option of marking records for deletion or erasing records.

Defining and loading a data set. You define all VSAM data sets in a catalog and allocate space for them by way of Access Method Services, rather than by way of JCL. You can load records into a data set with your own processing program or with Access Method Services, in one execution or in stages. Access Method Services does not merge input data sets, but merges an input data set with an output data set.

VSAM Functions That Go Beyond ISAM

VSAM has capabilities that ISAM doesn't have:

Skip sequential access. You can process a key-sequenced data set sequentially and skip records automatically, as though you were using direct access.

Concurrent request processing. Processing is extended by concurrent sequential or direct requests, or both, each requiring that VSAM keep track of a position in the data set, by means of a single access-method control block and without closing and reopening a data set.

Addressed sequential access. You can retrieve and store the records of a key-sequenced data set by RBA, as well as by key. With ISAM, you can position by physical address, but you must retrieve in a separate request.

Direct retrieval by generic key. With VSAM, you can retrieve a record directly, not only with a full-key search argument, but also with a generic search argument. ISAM enables you only to position at a record by generic argument: you must retrieve the record separately.

Secondary allocation of storage space. When you define a VSAM data set, you

can specify the amount of auxiliary-storage space that is to be allocated automatically, when required, beyond the primary space allocation. You can specify the amount in terms of a number of data records or in terms of a number of tracks or cylinders.

Automatic data-set reorganization. VSAM partially reorganizes a key-sequenced data set by splitting a control area when it has no more free control intervals and one is needed to insert a record.

No abnormal terminations by Open. The VSAM Open routine does not abnormally end, but returns an explanatory message in all cases where it cannot carry out a request to open a data set.

How to Convert an Indexed Sequential Data Set to a Key-Sequenced Data Set

To convert an indexed sequential data set to a VSAM data set that you can process either with an ISAM program by way of the ISAM interface or with a VSAM program, you must convert the ISAM JCL to VSAM JCL and use Access Method Services to define a key-sequenced data set and its index in a VSAM catalog and allocate space for them. You may use your ISAM load program by way of the ISAM interface to convert the data set, or you may use Access Method Services REPRO. For more details about the procedure, see the discussion of the Access Method Services DEFINE and REPRO commands in the section "How Is Access Method Services Used?" in the chapter "Communicating with VSAM."

Figure 11 summarizes converting indexed sequential data sets to key-sequenced data sets with indexes and processing them either with programs that have been converted from ISAM to VSAM or with programs that still use ISAM. Most existing programs that use ISAM require little or no modification to use the ISAM interface to process VSAM data sets.

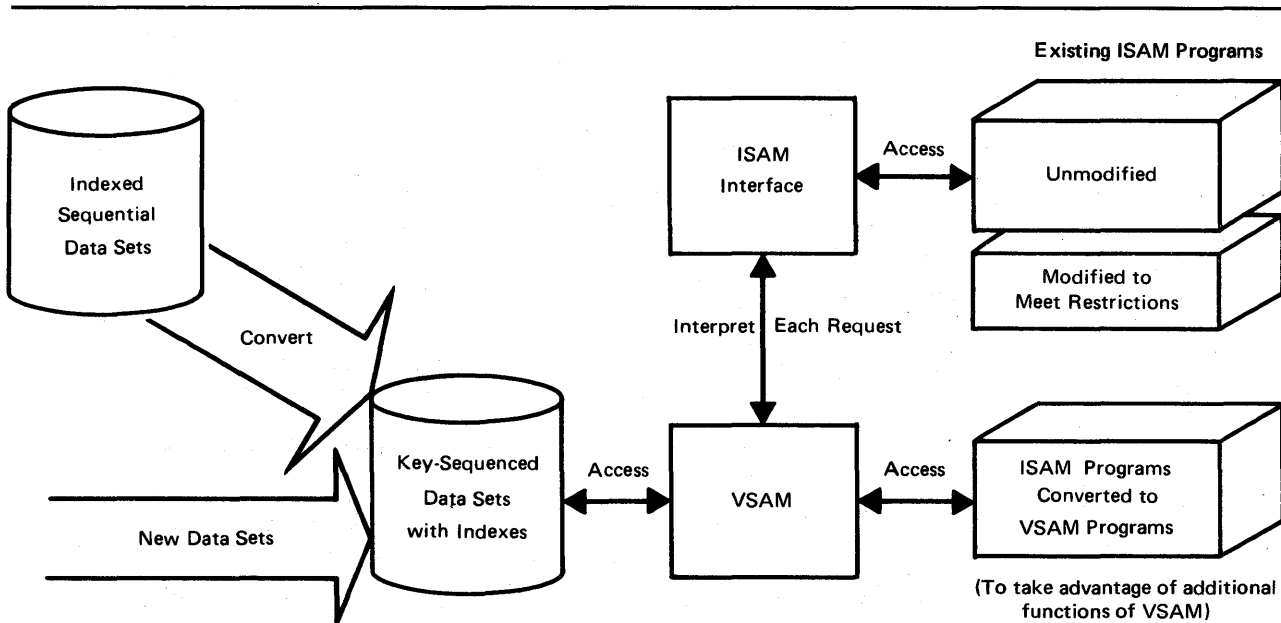


Figure 11. Use of ISAM Programs to Process VSAM Data Sets

What the ISAM Interface Does

When a processing program that uses ISAM (through assembler-language macros, PL/I, or COBOL) issues an OPEN that specifies a DCB describing an indexed-sequential data set that has been converted to or replaced by a key-sequenced data set, the Open routine detects the need for the ISAM interface and calls the interface's Open routine to:

- Construct control blocks and parameter lists that are required by VSAM
- Load the appropriate interface routines into virtual storage
- Initialize the ISAM DCB for the interface to intercept ISAM requests
- Take any DCB exit requested by the processing program

The interface intercepts each subsequent ISAM request, analyzes it to determine the equivalent keyed VSAM request, defines the keyed VSAM request in the request parameter list constructed by Open, and initiates the request.

All VSAM requests are handled synchronously; no VSAM CHECK macro is used. The ISAM ESETL and CHECK macros do not result in a VSAM macro's being issued. CHECK merely causes exception codes in the DECB (data event control block) to be tested.

For processing programs that use locate processing, the interface constructs buffers to simulate locate processing.

For blocked-record processing, the ISAM interface simulates unblocked-record processing by setting the overflow-record indicator for each record. (In ISAM, an overflow record is never blocked with other records.) The ISAM RELSE instruction causes no action to take place.

The interface receives return codes and exception codes for logical and physical errors from VSAM, translates them to ISAM codes, and routes them to the processing program or error-analysis (SYNAD) routine by way of the ISAM DCB or DECB.

When the processing program closes the data set, the interface's Close routine issues a VSAM PUT macro for ISAM PUT locate requests (in load mode), deletes from virtual storage the interface routines loaded by Open, frees virtual-storage space that was obtained by Open, and gives control to VSAM Close.

Restrictions in the Use of the ISAM Interface

The ISAM interface enables programs that use ISAM to issue only those requests that VSAM or the interface can simulate. These are the restrictions for using the interface:

- The program must run successfully under ISAM. The ISAM interface does not check for parameters that are invalid for ISAM.
- The OPENJ macro is not allowed; if your program uses it, remove it or replace it with the OPEN macro.
- If your program counts overflow records to determine reorganization needs, the count will be meaningless with VSAM data sets.
- You may share data among subtasks that specify the same DD statement in their DCB(s). But among subtasks that specify different DD statements for the data, you are responsible for data integrity. The ISAM interface doesn't

ensure DCB integrity when two or more DCBs are opened for a data set. Not all of the fields in a DCB can be counted on to contain valid information.

- The work area into which data records are read cannot be shorter than a record. If your processing program is designed to read a portion of a record into a work area, you must change the design. The record length in the DECB is assumed to be the actual length of the record.
- You cannot reload or reuse a key-sequenced data set as you can an indexed-sequential data set. You must use Access Method Services to delete the data set and redefine it.
- If your processing program issues the SETL I or SETL ID instruction, you must modify the instruction to some other form of the SETL or remove it. The ISAM interface cannot translate a request that depends on a specific block or device address.
- If your processing program was coded on the assumption that the indexed-sequential data set it was processing was a temporary data set, you may need to modify the program: a VSAM data set cannot be temporary.
- If the RECFM parameter is not specified in a processing program's DCB, you must specify it in the AMP parameter in the DD statement for the data set.
- A SYNAD routine must not issue VSAM macros or check for VSAM return codes. The ISAM interface translates all VSAM codes to appropriate ISAM codes. If your processing program already indicates a SYNAD routine, the routine specified in the AMP SYNAD parameter replaces it. You need not modify or replace a SYNAD routine that issues only a CLOSE, ABEND, SYNADAF, or SYNADRLS macro or examines DCB or DECB exception codes.

OPTIMIZING THE PERFORMANCE OF VSAM

This chapter is intended for programmers who will choose and implement the VSAM options that affect performance through the size of the control interval, the percents of distributed free space, and the handling of indexes and VSAM catalogs.

How Can Control-Interval Size Be Used to Influence Performance?

A data set's control-interval size affects performance. As a general rule the larger the control interval, the better the performance—for a number of reasons:

- Fewer index records required for a key-sequenced data set
- Fewer control-interval accesses, which is significant only for sequential or skip sequential access
- More efficient distribution of free space in a key-sequenced data set

You can request a particular control-interval size, but it must fall within the acceptable limits determined by VSAM, depending on the smallest amount of virtual-storage space you'll ever provide for I/O buffers and the size of your data records.

I/O-buffer size is important because VSAM transmits the contents of a control interval, and the amount of virtual-storage space for I/O buffers limits the size of a control interval. The amount of space for I/O buffers is the most flexible variable you have for influencing performance through control-interval size. The size and other attributes of your data records generally depend on the needs of your application.

How Does Distributed Free Space Improve Performance?

In the section "The Use of Free Space for Processing a Key-Sequenced Data Set" under "Keyed Access for Key-Sequenced Data Sets" in the chapter "Getting to Know What VSAM Is and Does," we discussed the way VSAM uses distributed free space for the insertion of a record into a key-sequenced data set. It was pointed out that insertion can be achieved in a data set that hasn't any distributed free space, by means of a control-area split. Therefore, the decision to provide free space throughout the control intervals and control areas of a data set rests on considerations of performance. Free space in the immediate area into which a record is inserted speeds up the insertion and avoids control-area splitting, which may move a group of records to a different cylinder, away from the preceding and following records in key sequence.

The question that arises is: How much space do I provide? There is no one answer; the decision depends on how much inserting or lengthening of records you plan to do. Of course, if the data set will be for reference only, it will need no free space. If insertions into the data set are numerous, you might get the best performance by leaving half of the space free when you create the data set. In general, you should estimate the percent of growth and leave a proportionate amount of free space. For example, if you calculated 25% growth, you might leave 1/5 of the total space free, because the data set is now at 4/5 of its eventual size.

You may estimate that the growth of a data set will continue indefinitely. But if you attempted to leave enough free space for indefinite growth, you would end up with almost nothing but free space. So you have to decide how long a period of growth you want to provide for and count on reorganizing the data set at the end of that period to redistribute free space.

When you estimate data-set growth, remember that if records in a key-sequenced data set are deleted or shortened, VSAM makes the space thus freed available as free space.

What Index Options Are There to Improve Performance?

Four options influence performance through the use of the index with a key-sequenced data set. Each option improves performance, but some of them require that you provide additional virtual- or auxiliary-storage space. The options are:

- Index-set records in virtual storage
- Index and data set on separate volumes
- Sequence-set records adjacent to the data set
- Replication of index records

Index-Set Records in Virtual Storage

To retrieve a record from a key-sequenced data set or store a record in it using keyed access, VSAM needs to examine the index of that data set. Before your processing program begins to process the data set, it must specify the amount of virtual-storage space it is providing for VSAM to buffer index records. Enough space for one I/O buffer for index records is the minimum, but a serious performance problem with a space large enough for only one or two index records is that an index record may be continually deleted from virtual storage to make room for another and then retrieved again later when it is required. Ample space to buffer index records can improve performance by preventing this situation.

You ensure that index-set records will be in virtual storage by specifying enough virtual-storage space for I/O buffers for index records when you begin to process a data set. VSAM keeps as many index-set records in virtual storage as the space will hold. Whenever an index record must be retrieved to locate a record, VSAM makes room for it by deleting from the space the index record that VSAM judges to be the least useful under the circumstances then prevailing. It is generally the index record that belongs to the lowest index level then represented in the space and that has been in the space the longest.

Index and Data Set on Separate Volumes

You may place the index of a key-sequenced data set on a separate volume from the data set, either on the same or on a different type of storage device.

Using different volumes eliminates the contention between gaining access to index records and gaining access to data records when you are using keyed access. The smaller amount of auxiliary-storage space required for an index makes it economical to use a faster storage device for it than for the data set.

Sequence-Set Records Adjacent to the Data Set

In using disk storage, you should minimize disk-arm movement. Having the sequence set accompany the data set is one way to reduce the movement for a key-sequenced data set. When you define the data set, you can specify that the sequence-set index record for each control area is to be on the first track of the control area. This avoids two separate seeks when access to a data record requires VSAM to examine the sequence-set index record of the control area in which the data record is stored. One arm movement enables VSAM to retrieve or store both the index record and the contents of the control interval in which the data record is stored. When this option is taken, sequence-set records are replicated, as described next.

Replication of Index Records

The last option is the replication of an index record on a track of a direct-access storage volume as many times as it will fit. The object of replication is to reduce the time lost waiting for the record to come around to be read (rotational delay). Rotational delay is, on the average, half the time it takes for the volume to rotate. Replication of a record reduces this time. For instance, if ten copies of an index record fit on a track, rotational delay is, on the average, only one-twentieth of the time it takes for the volume to rotate.

This option costs auxiliary-storage space; it requires a full track of storage for each index record replicated. You have to weigh the relative values of auxiliary-storage space and processing speed.

You can replicate index records in these combinations of sequence set and index set:

- Sequence set separated from index set and only sequence-set records replicated
- Sequence set separated from index set but all index records replicated
- Sequence set and index set together and all index records replicated

Separating the sequence set from the index set is for placing the sequence set adjacent to the data, which is the previous option we discussed. Figure 12 illustrates replication of a sequence-set record that has been placed adjacent to its control area to avoid moving the arm separately for index and for data; the index record is replicated to reduce rotational delay.

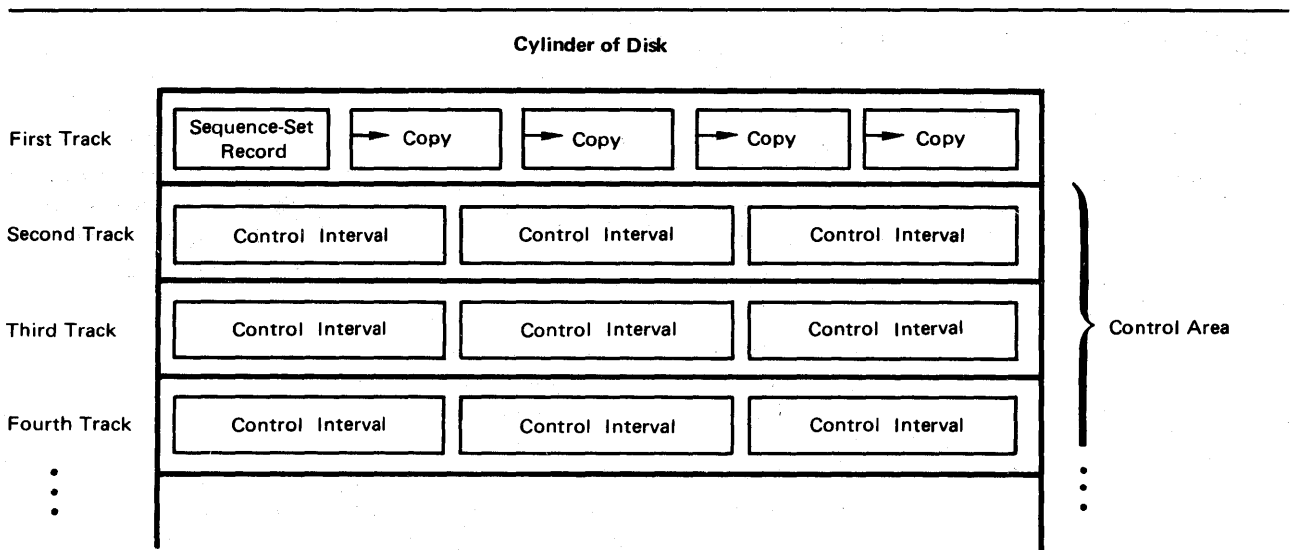


Figure 12. Replication of a Sequence-Set Index Record Adjacent to Its Control Area

How Can VSAM Catalogs Affect Performance?

Both the required master catalog and optional user catalogs can be used to improve performance.

Searching a VSAM Catalog

Because a VSAM catalog is organized as a key-sequenced data set with an index, searching it is faster than searching the system catalog. For this reason, VSAM catalogs are searched before the system catalog, for both VSAM data sets and data sets of other access methods. You can improve the performance of catalog-information retrieval by cataloging in a VSAM catalog not only your VSAM data sets, but also data sets of other access methods, except data sets that belong to generation data groups.

Sharing Services with User Catalogs

User catalogs are useful for improving performance. By putting the catalog information of some of your data sets and storage volumes into user catalogs, you reduce the search time for a given catalog and reduce the contention for the services of the master catalog.

PROTECTING DATA WITH VSAM

How safe is your data with VSAM? What provisions does VSAM make to ensure that data is not lost or destroyed by errors in the system, or sabotaged or pilfered by unauthorized persons? How easy is it to determine what the cause of a problem is and to do something about it? This chapter is intended for installation managers and system programmers interested in the answers to these questions.

The protection of data includes data integrity, or the safety of data from accidental destruction, and data security, or the safety of data from theft or intentional destruction. We'll discuss the attributes and options of VSAM that enhance data integrity, procedures for providing a backup copy of the catalog to protect data sets in the event of a catalog failure, protection of data shared by operating systems, regions, and subtasks, use of passwords and various authorization routines to prevent unauthorized access to your data, and methods of restart and problem determination.

How Does VSAM Achieve Data Integrity?

The attributes and options of VSAM that affect data integrity are:

- Method of inserting records into a key-sequenced data set
- Control-interval principle
- Method of indicating the end of a data set
- Verifying write operations

Method of Inserting Records into a Key-Sequenced Data Set

We discussed the method of inserting new records into a key-sequenced data set with an index in the section "The Use of Free Space for Processing a Key-Sequenced Data Set" under "Keyed Access for Key-Sequenced Data Sets" in the chapter "Getting to Know What VSAM Is and Does." Free space distributed throughout used control intervals allows VSAM to insert a record into a control interval held in virtual storage by shifting records in it without an I/O operation. VSAM splits control intervals and control areas, when necessary, in a way that does not expose any data to loss, even if an I/O error occurs before the split is completed.

Control-Interval Principle

With a key-sequenced data set, the control interval is the unit pointed to by entries in a sequence-set index record. Only a record addition or a record insertion that splits a control interval or a control area causes a modification of the index. For instance, even though a record insertion might change the RBA of the record with the highest key in the control interval, the index entry is not altered, since the pointer in it is to the control interval, not to the record. Minimal index handling and modification lessen the chance of error.

Method of Indicating the End of a Data Set

VSAM combines two procedures for achieving data integrity:

- Preformatting the last control area of a data set
- Updating the catalog to indicate the RBA of the end of the data set and the highest-keyed record in the data set

Preformatting a Data Set

Preformatting the end of a data set as each control area comes into use ensures greater data integrity than formatting it only at the end of processing. VSAM formats a control area before using its control intervals by putting control information in them and putting an end-of-file indicator in the last control interval. The end-of-file indicator helps prevent data that has been added to a data set from being lost.

VSAM optionally preformats control areas when loading records into a data set and always preformats them when subsequently adding records to the data set. You have two options when loading records into a data set, whether you use the REPRO command of Access Method Services or your own processing program:

- The first option is to improve load speed: VSAM does not format the last control area of a data set until a CLOSE macro instruction is issued. An error that prevents further processing will result in the loss of all of the data that has been loaded.
- The second option is to improve the ability to recover from a failure and complete loading. Each time a control area is filled with records, VSAM formats the next control area before storing records in it. In this way each set of new records is protected against loss as it is added to the data set.

Updating the Catalog

The addresses kept by the catalog for the end of the data set enable VSAM to keep track of the physical end and, for a key-sequenced data set, the logical end of the data set. VSAM updates these addresses at intervals determined by a processing program's issuance of a temporary CLOSE macro instruction and at the end of data-set processing, when the data set is fully closed. By using the VERIFY command of Access Method Services, you can recover data in cases where VSAM was unable to close a data set properly and update the end-of-file indicator in the catalog. See the discussion of the VERIFY command in the chapter "Communicating with VSAM."

Verifying Write Operations

To improve the integrity of data written to auxiliary storage, you can request VSAM to verify each write operation for accuracy. Verification takes additional time, but it decreases the chance of introducing errors into the data set.

Protecting the Catalog

The IMPORT and EXPORT Access Method Services commands cannot be used to back up VSAM catalogs. To protect your data sets and catalog in the event of a catalog failure, you may:

- Periodically dump the catalog and related data sets, using IEHDASDR or an independent utility program. In the event of failure, the backup copies can be restored, and data-set records can be updated by rerunning the jobs that were run since the backup copies were made.
- Periodically use EXPORT to create backup data sets. If data sets are exported after each job, restoration to current catalog status requires only that data sets be imported.
- Periodically dump the catalog, using IEHDASDR or an independent utility. In the event of failure, the backup copy of the catalog will be restored to the same physical location. Data-set records can then be updated by reference to SMF records to determine any changes since the backup copy was made and by using a user-written program that uses Access Method Services and/or the HMASPZAP service aid program.

To minimize the problem associated with catalog failure and the subsequent need to recover, use user catalogs widely, and, if possible, put the user catalogs on dedicated volumes. If you do so, the number of transactions per catalog is greatly reduced in the event of failure, and, therefore, the task of restoring the catalog is minimized.

How Is Shared Data Protected?

Data can be shared by different operating systems, by different jobs in a single system, and by different subtasks in an address space. There are provisions for controlling data sharing, and, therefore, protecting the integrity of data.

In determining the level of sharing you intend to allow, you must evaluate the consequences of a loss of *read integrity* (reading the correct information) to the processing program and a loss of *write integrity* (writing the correct information) to the data-set owner.

When a read request is issued, VSAM reads an entire control interval into storage. VSAM forces exclusive control over a control interval when the control interval is being modified.

Data-set sharing is controlled by the interaction of:

- The use of the share option (SHAREOPTIONS parameter) in the Access Method Services DEFINE command; the share options are specified for single systems and multiple systems.
- The use of the SHR and OLD parameters in the DD statement that identifies the data set.
- The type of processing (input or output) for which the data set was opened.
- The use of the RESERVE macro with shared direct-access storage devices.
- The use of the ENQ and DEQ macros.

If a data set cannot be shared and is not available when you request to open it, the request is denied.

When you issue the OPEN macro for an access-method control block, the Open routine enqueues (but does not reserve for exclusive use) the names of the components of a cluster—a cluster is made up of an entry-sequenced data set or of a key-sequenced data set and its index. If DISP=OLD is specified in a DD statement, only the *dsname* associated with that DD statement is exclusively reserved by the operating system. That is, only the cluster name is reserved. To have the cluster component(s) reserved as well (to avoid having one of them unavailable), you may include DD statements with DISP=OLD for the component(s) of the cluster. This practice will ensure that all resources needed to open the data set will be exclusively reserved before your task is initiated.

Subtask Sharing

Subtasks within a region may share a data set through a single DD statement or through separate DD statements.

When separate DD statements are used and one or more subtasks is to perform output processing, the DD statements must specify DISP=SHR. With separate DD statements, several subtasks can share a data set under the same rules as cross-region sharing.

With a single DD statement, several subtasks can update a data set concurrently. Subtask sharing with a single DD statement is independent of the DISP specification. If, however, DISP=SHR is specified, subtask sharing and cross-region sharing can occur concurrently.

To update a record, VSAM forces exclusive control over the control interval in which the record is stored.

A GET macro, issued for update, gains exclusive control over the control interval in which the record to be updated is stored. When a subtask has exclusive control of a control interval, a GET macro, issued for update, against the same control interval by another subtask is refused, but may be reissued later. Exclusive control is relinquished when any request is made for data that is outside the control interval or when an ENDREQ macro is issued.

A read request may be satisfied for a resource that is being shared for both read and update processing. An update request may be satisfied for a resource that is being shared for read processing only.

Cross-Region Sharing

Independent job steps in a system may request the use of a data set at the same time. To share a data set, each job step must specify a disposition of SHR in its DD statement for the data set. The type of sharing allowed depends on the data set's share attribute in the catalog.

The following share options apply in a single-system environment:

- The data set may be shared by any number of users for input processing *or* used by one user for output processing. Full integrity is maintained with this option.
- The data set may be used by any number of users for input processing *and* by one user for output processing. Write integrity is maintained with this option; the user, however, must assume responsibility for read integrity.

- The data set may be fully shared, but the user must assume full responsibility for read and write integrity; VSAM does nothing to assure integrity.
- The data set may be fully shared. Buffers used for direct processing are refreshed for each request. This option allows you to use the ENQ and DEQ macros to maintain data integrity while sharing the data set.

Cross-System Sharing

The following share options, which you may specify when you define a data set, apply in a multiple-system environment:

- The data set may be fully shared, but the user must assume full responsibility for read and write integrity.
- The data set may be fully shared, and buffers used for *direct processing* are refreshed for each request. The RESERVE and RELEASE macros may be used in conjunction with this option to maintain data integrity while sharing the data set.

Job steps of two or more OS/VS systems may gain access to the same data set regardless of the disposition specified in each step's JCL. To get exclusive control of a volume, a task in one system must issue a RESERVE macro.

Note: In a *shared-DASD* environment, integrity cannot be guaranteed by the system when users are allowed to share a data set for both input processing and output processing. VSAM does, however, provide assistance in protecting the integrity of the catalog.

How Can Passwords Be Used to Authorize Access?

Passwords are optional: you do not have to have them to gain access to a data set. But for added security, you can define passwords for data sets, indexes, and VSAM catalogs. There are different passwords for various degrees of data integrity:

- Full access. This is the master password, which allows you to gain access to a data set and any index and catalog record associated with it for all operations (retrieving, updating, inserting, deleting). Using this password to gain access to a catalog record gives you the ability to delete an entire data set and to alter password information or any other information in the catalog about a data set, index, or catalog.
- Control-interval access. This password authorizes you to use control-interval access and to override timestamp mismatches.
- Update access. This password authorizes you to retrieve, update, insert, or delete records in a data set. It gives you limited access to catalog records: you can define objects and alter their definitions, but you cannot delete entries.
- Read access. This is the read-only password, which allows you to examine data records and catalog records, but not to add, alter, or delete them.

The passwords associated with a data set, index, or catalog are specified through Access Method Services when you define it. This information is kept in the catalog, and when a processing program attempts to open a data set, the security-verification routine checks whether a password is required and whether the correct one is given. Computer operators and

communications-terminal users may also be given the opportunity to supply the correct password, and you can specify how many times they may try to do so.

Besides VSAM password protection, you may also have your own routine to check a requester's authority. You can define security-authorization records in the master catalog or in a user catalog to contain whatever special password information you wish, for use by your authorization routine. VSAM transfers control to your routine when a requester gives a correct password other than the master password.

How Are Programs Restarted Following a Failure?

In general, the checkpoint/restart program for VSAM data sets is similar to that provided by OS/VS for ISAM and BDAM.

Recording Checkpoint Information

To restart after a failure that terminated processing, it is necessary to determine the status of processing programs when the failure occurred. A processing program defines a checkpoint by issuing a CHKPT macro instruction. The checkpoint program issues a VSAM temporary CLOSE macro to update the catalog. It then records information about VSAM data sets in a checkpoint data set. If a failure occurs, the latest checkpoint record can be used to reconstruct the situation that prevailed before the failure.

Restarting the Processing Program

Restart is the procedure of processing the checkpoint record and giving control back to the processing program interrupted by the failure. Different types of restart are distinguished for VSAM, for:

- Entry-sequenced output data sets. An entry-sequenced output data set is restored by the elimination of all records that have been added at the end since the checkpoint.
- Input data sets or key-sequenced data sets open for addressed access. A data set that was open for input at the checkpoint or a key-sequenced data set that was open for addressed access is prepared for restart by the restoration of any statistical information (such as number of records inserted) to its checkpoint status.

Restrictions and Options for Restarting a Program

The VSAM DD parameter, AMP, has a subparameter for specifying checkpoint/restart options that handle two special situations in restarting a processing program:

- Modifications other than records added sequentially to the end of a data set. The restart program cannot restore a data set to its checkpoint status if there have been internal modifications to it since the checkpoint, and the restart program will normally not attempt to restart processing.
- Addition of records to the end of a data set by way of a job step other than the job step that issued the checkpoint. Any records added to the end of a data set will normally be erased in restoring the data set to its checkpoint status.

The AMP options for checkpoint/restart are: to let restart take its normal action for either situation, to override either one or the other of the two actions, or to override both. If you override the check for internal modification, your processing program is restarted, even though the data set it was processing cannot be restored; if you override the erasure of data at the end of a data set, your processing program is not restarted, unless you also override the check for modification.

A third situation that restart may encounter is a data set that was not closed following the failure and that cannot be restarted because of the resulting loss of end-of-file information in the catalog. You need to use the VERIFY command of Access Method Services to enable you to restart processing in such a case.

For more information about checkpoint/restart with OS/VS, see *OS/VS Checkpoint/Restart*, GC26-3784.

How Can the Causes of Problems Be Determined?

VSAM offers several diagnostic aids for you to determine what's wrong when things don't work.

Exits to Your Error-Analysis Routines

VSAM provides two optional exits to routines you supply to handle error situations. If you provide the exit routines for analyzing errors, your processing program can investigate many errors and decide what to do in an orderly manner. Not only physical errors, but also logical errors that may arise out of unlikely combinations of events in a complex application can be handled by exits.

VSAM Messages

The messages put out by VSAM for the operator and programmer are designed to help them understand both the nature of the problem and the exact steps to take to correct it. Other messages that originate with VSAM are the diagnostic messages that are made available to your physical-error analysis routines in a special message area provided by your processing program.

Generalized Trace Facility (GTF)

GTF is an optional program of OS/VS that continually records, as they occur, events of selected classes that are necessary to trace a processing program. You must weigh the relative values of this diagnostic ability and the added processing time required. It is a debugging tool and a maintenance aid: it produces unformatted output. To format and print this output, use the Edit function of the HMDPRDMP or AMDPRDMP service aid. For information about GTF or the Edit function, see *OS/VS Service Aids*, GC28-0633.

GLOSSARY

The following terms are defined as they are used in this book. If you do not find the term you are looking for, refer to the index or to the *IBM Data Processing Glossary, GC20-1699*.

Access Method Services: A multifunction service program that defines VSAM data sets and allocates space for them, converts indexed sequential data sets to key-sequenced data sets with indexes, modifies data-set attributes in the catalog, reorganizes data sets, facilitates data portability between operating systems, creates backup copies of data sets and indexes, helps make inaccessible data sets accessible, and lists data-set records and catalog records.

addressed direct access: The retrieval or storage of a data record identified by its relative byte address, independent of the record's location relative to the previously retrieved or stored record. (*See also* keyed direct access, addressed sequential access, and keyed sequential access.)

addressed sequential access: The retrieval or storage of a data record in RBA sequence relative to the previously retrieved or stored record. (*See also* keyed sequential access, addressed direct access, and keyed direct access.)

application: As used in this publication, the use to which an access method is put or the end result that it serves; contrasted to the internal operation of the access method.

catalog: (*See* master catalog and user catalog.)

collating sequence: An ordering assigned to a set of items, such that any two sets in that assigned order can be collated. As used in this publication, the order defined by the System/370 8-bit code for alphabetic, numeric, and special characters.

compression: (*See* key compression.)

control area: A group of control intervals used as a unit for formatting a data set before adding records to it. Also, in a key-sequenced data set, the set of control intervals pointed to by a sequence-set index record; used by VSAM for distributing free space and for placing a sequence-set index record adjacent to its data.

control-area split: The movement of the contents of some of the control intervals in a control area to a newly created control area, to facilitate the insertion or lengthening of a data record when there are no remaining free control intervals in the original control area.

control interval: A fixed-length area of auxiliary-storage space in which VSAM stores records and distributes free space. It is the unit of information transmitted to or from auxiliary storage by VSAM, independent of physical record size.

control-interval access: The retrieval or storage of the contents of a control interval.

control-interval split: The movement of some of the stored records in a control interval to a free control interval, to facilitate the insertion or lengthening of a record that won't fit in the original control interval.

data integrity: Preservation of data or programs for their intended purpose. As used in this publication, the safety of data from inadvertent destruction or alteration.

data record: A collection of items of information from the standpoint of its use in an application and not from the standpoint of the manner in which it is stored (*see also* stored record).

data security: Prevention of access to or use of data or programs without authorization. As used in this publication, the safety of data from unauthorized use, theft, or purposeful destruction.

data set: The major unit of data storage and retrieval in the operating system, consisting of data in a prescribed arrangement and described by control information to which the system has access. (*See also* key-sequenced data set and entry-sequenced data set.)

data space: A storage area defined in the volume table of contents of a direct-access volume for the exclusive use of VSAM to store data sets, indexes, and catalogs.

direct access: The retrieval or storage of data by a reference to its location in a data set rather than relative to the previously retrieved or stored data. (*See also* addressed direct access and keyed direct access.)

distributed free space: Space reserved within the control intervals of a key-sequenced data set for inserting new records into the data set in key sequence; also, whole control intervals reserved in a control area for the same purpose.

entry sequence: The order in which data records are physically arranged (according to ascending RBA) in auxiliary storage, without respect to their contents. (Contrast to key sequence.)

entry-sequenced data set: A data set whose records are loaded without respect to their contents, and whose relative byte addresses cannot change. Records are retrieved and stored by addressed access, and new records are added at the end of the data set.

extent: A continuous space allocated on a direct-access storage volume, reserved for a particular data space or data set.

field: In a record or a control block, a specified area used for a particular category of data or control information.

free space: (*See* distributed free space.)

generic key: A leading portion of a key, containing characters that identify those records that are significant for a certain application. For example, it might be desirable to retrieve all records whose keys begin with the generic key AB, regardless of the full key values.

horizontal pointer: A pointer in an index record that gives the location of another index record in the same level that contains the next key in collating sequence; used for keyed sequential access.

index: As used in this publication, an ordered collection of pairs, each consisting of a key and a pointer, used by VSAM to sequence and locate the records of a key-sequenced data set; organized in levels of index records. (*See also* index level, index set, and sequence set.)

index entry: A key and a pointer paired together, where the

key is the highest key (in compressed form) entered in an index record or contained in a data record in a control interval, and the pointer gives the location of that index record or control interval.

index level: A set of index records that order and give the location of records in the next lower level or of control intervals in the data set that it controls.

index record: A collection of index entries that are retrieved and stored as a group.

index replication: The use of an entire track of direct-access storage to contain as many copies of a single index record as possible; reduces rotational delay.

index set: The set of index levels above the sequence set. The index set and the sequence set together comprise the index.

integrity: (*See* data integrity.)

ISAM interface: A set of routines that allow a processing program coded to use ISAM (indexed sequential access method) to gain access to a key-sequenced data set with an index.

key: As used in this publication, one or more consecutive characters taken from a data record, used to identify the record and establish its order with respect to other records. (*See also* key field and generic key.)

key compression: The elimination of characters from the front and the back of a key that VSAM does not need to distinguish the key from the preceding or following key in an index record; reduces storage space for an index.

key field: A field located in the same position in each record of a data set, whose contents are used for the key of a record.

key sequence: The collating sequence of data records, determined by the value of the key field in each of the data records. May be the same as, or different from, the entry sequence of the records.

key-sequenced data set: A data set whose records are loaded in key sequence and controlled by an index.

keyed direct access: The retrieval or storage of a data record by use of an index that relates the record's key to its relative location in the data set, independent of the record's location relative to the previously retrieved or stored record. (*See also* addressed direct access, keyed sequential access, and addressed sequential access.)

keyed sequential access: The retrieval or storage of a data record in its key sequence relative to the previously retrieved or stored record. (*See also* addressed sequential access, keyed direct access, and addressed direct access.)

mass sequential insertion: A technique VSAM uses for keyed sequential insertion of two or more records in sequence into a collating position in a data set: more efficient than inserting each record directly.

master catalog: A key-sequenced data set with an index containing extensive data-set and volume information that VSAM requires to locate data sets, to allocate and deallocate storage space, to verify the authorization of a program or operator to gain access to a data set, and to accumulate usage statistics for data sets.

password: A unique string of characters stored in a catalog

that a program, a computer operator, or a TSO terminal user must supply to meet security requirements before a program gains access to a data set.

physical record: A physical unit of recording on a medium, for example, the physical unit between address markers on a disk.

pointer: An address or other indication of location. For example, an RBA is a pointer that gives the relative location of a data record or a control interval in the data set to which it belongs. (*See also* horizontal pointer and vertical pointer.)

portability: The ability to use VSAM data sets with different operating systems. Volumes whose data sets are cataloged in a user catalog can be demounted from storage devices of one system, moved to another system, and mounted on storage devices of that system. Individual data sets can be transported between operating systems using Access Method Services.

random access: (*See* direct access.)

RBA: Relative byte address. The displacement of a data record or a control interval from the beginning of the data set to which it belongs; independent of the manner in which the data set is stored.

record: (*See* index record, data record, stored record.)

relative byte address: (*See* RBA.)

replication: (*See* index replication.)

security: (*See* data security.)

sequence set: The lowest level of the index of a key-sequenced data set; it gives the locations of the control intervals in the data set and orders them by the key sequence of the data records they contain. The sequence set and the index set together comprise the index.

sequential access: The retrieval or storage of a data record in either its entry sequence or its key sequence, relative to the previously retrieved or stored record. (*See also* addressed sequential access and keyed sequential access.)

skip sequential access: Keyed sequential retrieval or storage of records here and there throughout a data set, skipping automatically to the desired record or collating position for insertion: VSAM scans the sequence set to find a record or a collating position.

stored record: A data record, together with its control information, as stored in auxiliary storage.

user catalog: A catalog used in the same way as the master catalog, but optional and pointed to by the master catalog, and also used to lessen the contention for the master catalog and to facilitate volume portability.

vertical pointer: A pointer in an index record of a given level that gives the location of an index record in the next lower level or the location of a control interval in the data set controlled by the index.

INDEX

Indexes of OS/VS reference publications are consolidated in the *OS/VS Master Index*, GC28-0602. For additional information about any subject listed in this index, refer to the publications that are listed under the same subject in the master index.

This index makes no page references to the glossary.

A

ACB macro 37
access (*see* keyed access and addressed access)
access method, requirements for 12
Access Method Services
 ALTER command 32
 altering sequence of command execution 31
 checkpoint/restart 58
 DEFINE command 32
 DELETE command 33
 EXPORT command 34
 how used 31
 IMPORT command 34
 LISTCAT command 33
 PRINT command 33
 REPRO command 33
 summary of functions 31
 VERIFY command 31
access-method control block
 changing 40
 defining with ACB macro 37
 more than one with same DD statement 37
accessibility of data, testing 34
addressed access
 deletion with key-sequenced data set 25
 differences between entry- and key-sequenced data sets 25
 marking records inactive with entry-sequenced data sets 25
 positioning VSAM for subsequent access 25
 retrieval 25
 storage 25
addressed direct access 25
addressed sequential access 25
addressing data records 17
allocating space
 Access Method Services 27
 by range of key values 32
 comparison with ISAM 44
 independently of device 32
 on unmounted volumes 27
 restriction 27
ALTER command of Access Method Services 32
AMDPRDMP service aid 59
AMP JCL DD parameter
 checkpoint/restart 58
 general description 42
 ISAM interface 48
arm movement, minimizing 51
assembler language 43
asynchronous processing 38
attributes of a data set, changing 32
authorization to process a data set 57
auxiliary storage devices

 minimizing rotational delay 51
 space required for index replication 51
 VSAM can be used with 14

B

backing up a data set 34
basic direct access method (BDAM) 20
BDAM (basic direct access method) 20
beginning sequential access 22,25
buffer, I/O
 defining minimum space 50,32
 effect on performance 49
 index-set records resident in virtual storage 50
 specifying size and number 37

C

catalog (*see also* master catalog, user catalog, and system catalog)
 protecting 55
catalog record
 data set 28
 deleting 33
 listing 33
 modifying 32
 using a model to define 32
 volume 28
causes of problems, determining 59
central processing units (CPUs)
 models 14
 sharing data among 55
chaining request parameter lists 38
changes in relative byte address
 exit routine for recording 21,28
 key-sequenced data set 20
changing a record's length (*see* shortening a record and lengthening a record)
changing attributes of a data set
 by reorganizing data sets 32
 in catalog record 32
changing control blocks and lists 40
character elimination, in keys 19
CHECK macro 40
checking write operations for accuracy 54
checkpoint/restart
 recording checkpoint information 58
 restarting processing 58
 restrictions 58
 specifying in AMP JCL DD parameter 59
CHKPT macro 58
CLOSE macro
 disconnecting program from data 36
 indicating the end of a data set 54
 ISAM interface 47
COBOL language 43
collating sequence 15
 (*see also* key sequence)
combining data sets 33
commands of Access Method Services (*see* Access Method Services)
 (*see also* macros)
compression, key 19

- concatenated data sets, not allowed 42
 - concurrent request processing
 - definition 21
 - number of I/O buffers used in 37
 - protecting data during 56
 - specifying the number of requests 37
 - conditional statements, Access Method Services 31
 - configuration, system 14
 - connecting a user catalog to the master catalog 34
 - connecting program to data 36
 - control area
 - definition 16
 - preformatting 54
 - relation to control interval 16
 - relation to extent of data set 16
 - relation to sequence set 16,52
 - illustration 19
 - size 16
 - split 24
 - control block
 - access-method control block 37
 - changing 40
 - exit list 38
 - request parameter list 38
 - control information in stored record 17
 - control interval
 - definition 15
 - determining size 15,32
 - effect of size on performance 49
 - how it helps protect data 53
 - maximum size 17
 - number in a control area 16
 - relation to control area 16
 - size independent of physical record size 16
 - split 23
 - control-interval access
 - definition 21
 - specifying in the macros 38,40
 - conversational time sharing 43
 - converting data sets to VSAM format
 - indexed sequential data sets 46,33
 - REPRO command of Access Method Services 33
 - copying data sets 33
 - core (*see* virtual storage)
 - CPUs (central processing units)
 - models 14
 - sharing data among 55
 - creating a data set 31,21
 - cross-region sharing of data 56
 - cross-system sharing of data 57
- ## D
- DASDs (direct-access storage devices)
 - minimizing rotational delay 51
 - space required for index replication 51
 - VSAM can be used with 14
 - DAT (dynamic address translator) 14
 - data format 17
 - data integrity
 - checkpoint/restart 58
 - concurrent request processing 56
 - definition 13
 - determining the causes of problems 59
 - options 53
 - passwords 57
 - shared data 55
 - data management requirements for access method 12
 - data portability
 - data-set 34
 - illustration comparing data-set and volume portability 35
 - volume 34
 - data protection 13
 - (*see also* data integrity and data security)
 - data record
 - illustration 17
 - method of addressing 17
 - (*see also* relative byte address)
 - method of storing 17
 - restriction 17
 - data security
 - authorization routine 58
 - definition 13
 - passwords 57
 - data set
 - allocation 27,32
 - backup copy 34
 - catalog record 28
 - copying 33
 - defining 32
 - deleting 33
 - extents 16
 - illustration 16
 - indexed sequential 33
 - listing 33
 - maximum size 16
 - merging data sets 33
 - organization 15
 - partial volume mounting 42
 - preformatting 54
 - reorganizing 32
 - sequential 33
 - sharing 55
 - transporting 34
 - data space
 - allocation 32
 - definition 16
 - extents 16
 - illustration 17
 - data-set entry in catalog 28
 - data-set portability 34
 - DD statement 41,42
 - debugging 59
 - debugging tool (Generalized Trace Facility) 59
 - DEFINE command of Access Method Services 32
 - DELETE command of Access Method Services 33
 - deleting a catalog record 33
 - deleting a data set 33
 - deleting a record
 - addressed 25
 - changing relative byte addresses 20
 - comparison with ISAM 45
 - keyed 23
 - marking record inactive with entry-sequenced data set 25
 - reclamation of space 23
 - determining causes of problems 59
 - devices, auxiliary storage
 - minimizing rotational delay 51
 - space required for index replication 51
 - VSAM can be used with 14
 - diagnostic aids 59
 - direct access
 - addressed 25

- definition 21
- keyed 22
- matching search argument to key 22
- positioning for subsequent sequential access 22
- direct-access storage devices (DASDs)
 - minimizing rotational delay 51
 - space required for index replication 51
 - VSAM can be used with 14
- disconnecting a program from data 36
- disconnecting a user catalog from the master catalog 34
- disk storage
 - (*see also* direct-access storage devices)
 - minimizing arm movement 51
- distributed free space
 - distribution 19
 - effect on performance 49
 - estimating growth 49
 - for inserting records 23
 - protecting data 53
 - reclamation 19,23
- DOS/VS and OS/VS
 - data-set portability 34
 - volume portability 34
- dynamic address translator (DAT) 14

E

- end of data set, method of indicating 36
- end-of-data set processing 38
- end-of-file indicator
 - preformatting 54
 - updated by CLOSE 36
- ENDREQ macro 41
- entry (*see* catalog entry and index entry)
- entry sequence
 - affected by control-interval split 24
 - definition 15
- entry-sequenced data set
 - (*see also* data set)
 - definition 15
 - keeping track of relative byte addresses 20
- EODAD exit routine 38
- ERASE macro
 - addressed access 25
 - initiating access 40
 - keyed access 23
- erasing a data set 33
- erasing a record
 - addressed 25
 - changing relative byte addresses 20
 - comparison with ISAM 45
 - keyed 23
 - marking record inactive with entry-sequenced data set 25
 - reclamation of space 23
- error analysis 59,38
- error messages 59,40
- error-exit routine 38,59
- estimating data-set growth 49
- evaluating system usage with System Management Facilities 44
- examining control blocks and lists 40
- exclusive control for update 56
- exit list
 - changing 40
 - defining with the EXLST macro 38
- exit routines 38,59
- EXLST macro 38

- EXPORT command of Access Method Services 34
- extent
 - data set 16
 - data space 16
 - definition 16
 - relation to control area 16
- extracting catalog information for data portability 37

F

- failures, determining cause of 59
- fixed-head storage 14
- fixed-length records 17
- format of stored data 17
- formatting data set before storing records 54
- free space (*see* distributed free space)
- functions of VSAM 21

G

- GENCB macro 40
- Generalized Trace Facility (GTF) 59
- generating control blocks and lists 40
- generic key (partial key)
 - definition 22
 - searching for a match 22,39
- GET macro
 - initiating access 40
 - positioning 22,25
- getting a record
 - addressed 25
 - keyed 22
 - positioning 22,25
 - skipping 22
- growth, estimating data-set 49
- GTF (Generalized Trace Facility) 59

H

- high-level languages 43
- HMASPZAP service aid 55
- HMDPRDMP service aid 59
- horizontal pointer
 - definition 19
 - illustration 19
 - skip sequential access 22
- how Access Method Services is used 31
- how existing programs that use ISAM can be used with VSAM 44
- how programs are restarted following a failure 58
- how TSO can be used with VSAM 43

I

- IEHDASDR 55
- I/O buffer
 - defining minimum space 37,32
 - effect on performance 50
 - index-set records resident in virtual storage 50
 - relation to processing program work area 21,39
 - specifying size and number 37
- I/O errors 38
- IMPORT command of Access Method Services 34
- index
 - comparison with ISAM index 44
 - illustration 19
 - performance options 50
 - purpose 18

- requires minimal updating 53
- structure 18
- index entry
 - description 18
 - key compression 19
- index record
 - entries 18
 - kept in virtual storage 50
 - key compression 19
 - levels 18
 - replication 51
 - sequence-set record adjacent to control area 51
- index set
 - definition 18
 - description 18
 - illustration 19
 - physical placement in relation to sequence set 51
 - records resident in virtual storage 50
- indexed sequential access method (ISAM)
 - (see also indexed sequential data set and ISAM interface)
 - comparison with VSAM 44
- indexed sequential data set
 - converting to VSAM format 46,33
 - listing 33
- input/output buffer (see I/O buffer)
- inserting a record
 - changing relative byte addresses 20
 - control-area split 24
 - control-interval split 23
 - mass sequential insertion 23
 - protecting data 53
 - without split 23
- integrity of data
 - checkpoint/restart 58
 - definition 13
 - determining causes of problems 59
 - options 53
 - passwords 57
 - shared data 55
- interface (see ISAM interface)
- interpreting ISAM requests 47
- ISAM (indexed sequential access method)
 - (see also indexed sequential data set and ISAM interface)
 - comparison with VSAM 44
- ISAM data set (see indexed sequential data set)
- ISAM interface
 - converting data sets and job control language 46,33
 - operation 47
 - purpose 46
 - restrictions 47

J

- JCL (see job control language)
- job control language (JCL)
 - AMP DD parameter 42
 - defining a VSAM data set 41
 - processing a VSAM data set 41
 - restricted parameters 42
 - specifying VSAM catalogs 41
- JOBCAT JCL statement 41
- journalizing transactions 38

K

- key
 - allocating space on volumes by range 32
 - compression 19
 - generic (partial) 22
 - use in index 18
- key field
 - description 18
 - unique value 15
- key sequence
 - definition 15
 - sequence set 24
- key-sequenced data set
 - (see also distributed free space, index, and data set)
 - catalog structure 26
 - comparison with entry-sequenced data set 18
 - definition 15
 - keeping track of relative byte addresses 21
- keyed access
 - deletion 23
 - matching search argument to key 22
 - positioning 22
 - retrieval 22
 - skipping 22
 - space reclamation 23
 - storage 23
- keyed direct access 22
- keyed sequential access 22

L

- languages, programming 43
- lengthening a record
 - changing relative byte addresses 20
 - control-area split 24
 - control-interval split 23
 - entry-sequenced data set 26
 - without split 23
- levels of index
 - illustration 19
 - index set 18
 - sequence set 18
- LISTCAT command of Access Method Services 33
- listing
 - catalog records 33
 - data sets 33
- loading records into a data set
 - comparison with ISAM 45
 - preformatting options 54
 - REPRO command of Access Method Services 33
 - using a processing program 21,45
- locate processing
 - retrieval 22,39
 - simulation by ISAM interface 47
- logical record (see data record)
- logical-error analysis exit routine 38

M

machines VSAM can be used with
 central processing units 14
 storage devices 14

macros
 (*see also* Access Method Services for commands)
 ACB 37
 CHECK 40
 CHKPT 58
 CLOSE 36
 ENDREQ 41
 ERASE 23,25,40
 EXLST 38
 GENCB 40
 GET 22,25,40
 MODCB 40
 OPEN 36
 POINT 22,25,40
 PUT 23,25,40
 RESERVE 55
 RPL 38
 SHOWCB 40
 summary of VSAM macros 36
 TESTCB 40

main storage (*see* virtual storage)

making a data set portable 34

mass sequential insertion 23

master catalog
 (*see also* user catalog)
 cataloging nonVSAM data sets 26,52
 restriction 26
 how to improve reliability of 28
 indicating the end of a data set 54
 information in catalog records 27
 order of catalog search 26
 protecting 55
 relation to system and user catalogs 26

master password 57

maximum size of a control interval 17

maximum size of a data set 16

measuring system usage 44

memory (*see* virtual storage)

merging data sets 33

messages 59,40

method of indicating the end of a data set 54

MODCB macro 40

modifying a catalog record 32

modifying control blocks and lists 40

mounting only some volumes of a data set 42

moving data sets from one operating system to another 34

N

noting RBA changes 38

O

OPEN macro
 connecting program to data 36
 ISAM interface 47

operator entering passwords 57

optimizing the performance of VSAM 49

options
 in defining a data set 32
 in preformatting a data set 54
 in transporting data 34

performance 49
 types of access 21
 user catalogs 28

organization of a data set 15
 (*see also* data set)

OS/VSAM and DOS/VSAM
 data-set portability 34
 volume portability 34

overflow area
 (*see also* distributed free space)
 comparison with ISAM 45

P

parameter list
 exit list 38
 request parameter list 38

partial key (generic key)
 definition 22
 searching for a match 22,39

passwords 57

performance
 catalog 52
 general discussion 49
 improved by control-interval size 49
 improved by distributed free space 49
 index options 50
 illustration 52

permanent exportation 34

physical record (*see* stored record)

physical-error analysis
 exit routine 38
 ISAM interface 47

PL/I language 43

POINT macro
 addressed 25
 initiating access 40
 keyed 22

pointer
 catalog 28
 index 18

portability
 data-set 34
 illustration 35
 volume 34

positioning for sequential access
 by entry sequence 25
 by key sequence 22
 done by POINT macro 40
 with concurrent access 21

preformatting end of data set 54

PRINT command of Access Method Services 33

printing
 catalog entries 33
 data sets 33

problem analysis 59

processing types
 (*see also* keyed access and addressed access)
 specifying 37,38

program residence (VSAM routines)
 deleted by the Close routine 37
 illustration 11
 loaded by the Open routine 36

programming languages 43

protecting data 53
 (*see also* data integrity and data security)

PUT macro

- addressed 25
- initiating access 40
- keyed 23

Q

- QSAM (queued sequential access method) 20
- queued sequential access method (QSAM) 20

R

- random access (*see* direct access)
- ranges of key values for space allocation 32
- RBA (*see* relative byte address)
- reading a record
 - addressed 25
 - keyed 22
 - positioning 22,25
 - skipping 22
- reclamation of space
 - entry-sequenced data set 25
 - key-sequenced data set 23,20
- record
 - data record 17
 - index record 18
 - maximum size 17
 - stored record 17
- recording RBA changes 38
- regions sharing data 56
- relative byte address (RBA)
 - changeability in key-sequenced data set 20
 - definition 17
 - keeping track of
 - entry-sequenced data set 20,21
 - key-sequenced data set 21,38
 - unchangeability in entry-sequenced data set 18
- remote terminals 43
- reorganizing data sets
 - automatically through control-area split 46
 - by copying 33
 - comparison with ISAM 46
- replication of index records 51
- REPRO command of Access Method Services 33
- request parameter list
 - changing 32
 - defining with the RPL macro 38
- requesting access to a data set 40
- requirements
 - storage (*see* storage requirements)
 - system 14
- requirements for an access method 12
- RESERVE macro 55
- residence of VSAM routines
 - deleted by the Close routine 37
 - illustration 11
 - loaded by the Open routine 36
- restart 58
- retrieving a record
 - addressed 25
 - keyed 22
 - positioning 22
 - skipping 22
- reusing space in a data set
 - entry-sequenced data set 25
 - key-sequenced data set 23,20
- rotational delay, minimizing 51
- RPL macro 38

S

- SAM data set (*see* sequential data set)
- searching catalogs
 - order of 26
 - performance 52
- secondary storage (*see* auxiliary-storage devices)
- security of data
 - authorization routine 58
 - definition 13
 - passwords 57
- sequence set
 - definition 18
 - description 18
 - determining key sequence 24
 - physical placement in relation to index set 51
 - relation to control areas 18
- sequence-set records adjacent to the data set 51
- sequential access
 - addressed 25
 - definition 21
 - keyed 22
 - positioning 22,25
 - skipping 22
- sequential data set
 - converting 33
 - listing 33
 - form in which a VSAM data set is transported between systems 34
- sequential-access storage devices (*see also* sequential data set)
 - data-set transporting 34
- service aids 59
- service program (*see* Access Method Services)
- sharing data
 - between regions 56
 - between subtasks 56
 - between systems 57
- shortening a record
 - changing relative byte addresses 20
 - entry-sequenced data set 26
 - reclamation of space 23
- SHOWCB macro 40
- skip sequential access
 - definition 22
 - retrieval 22
 - storage 23
- SMF (System Management Facilities) 44
- space reclamation
 - entry-sequenced data set 25
 - key-sequenced data set 23,20
- special uses of user catalogs 28
- speed (*see* performance)
- split
 - control-area 24
 - control-interval 23
- STPCAT JCL statement 41
- storage devices
 - space required for index replication 51
 - VSAM can be used with 14
- storage requirements
 - free space 49
 - I/O buffers 37
 - index options 50
 - work areas 39
- stored record

- definition 17
- illustration 17
- maximum size 17
- storing a record
 - addressed 25
 - control information describing a record 17
 - keyed 23
 - mass sequential insertion 23
 - skipping 23
- substituting processing parameters by way of JCL 42,48
- subtasks sharing data
 - (*see also* concurrent request processing)
 - protection 56
- SYNAD exit routine
 - specifying the exit 38
 - using ISAM interface 48
- synchronizing asynchronous requests 40
- synchronous processing 39
- system catalog
 - order of search 26
 - points to master catalog 26
 - relation to master and user catalogs 26
- System Management Facilities (SMF) 44
- system requirements 14
- system usage evaluation with System Management Facilities 44
- System/370
 - models 14
 - sharing data among central processing units 57
- systems sharing data 57

T

- tape storage
 - (*see also* sequential data set)
 - data-set transporting 34
- tasks sharing data 56
- temporary CLOSE macro
 - functions 37
 - indicating end of data set 54
- temporary data sets, not allowed 42
- temporary exportation 34
- terminals 43
- terminating a request before completion 41
- TESTCB macro 40
- testing control blocks and lists 40
- Time Sharing Option (TSO) 43
- tracing 59
- translating ISAM requests 47
- transporting data between systems
 - data-set portability 34
 - illustration 35
 - volume portability 34
- TSO (Time Sharing Option) 43

U

- updating a record (*see* storing a record, lengthening a record, and shortening a record)
- usage, evaluating system, with System Management Facilities 44
- use of free space for processing a key-sequenced data set 23
- user catalog
 - (*see also* master catalog)
 - connecting to master catalog 34
 - disconnecting from master catalog 34
 - job control language 41

- order of search 26
- protecting 55
- reducing contention for master catalog 52
- volume portability 34
- utility program (*see* Access Method Services)

V

- variable-length records 17
- verification routine, security 58
- VERIFY command of Access Method Services 31
- verifying write operations 54
- vertical pointer
 - definition 18
 - illustration 19
 - keyed direct access 22
- virtual storage
 - dynamic address translator 14
 - index records kept resident 50
 - (*see also* I/O buffer)
- Virtual Storage Access Method (VSAM)
 - comparison with indexed sequential access method 44,45
 - requirements for data processing 12
- volume record in catalog 28
- volume portability 34
- VSAM (Virtual Storage Access Method)
 - comparison with indexed sequential access method 44
 - requirements for data processing 12

W

- work area
 - relation to I/O buffer 21
 - specifying 39
- write operation, verification 54
- writing a record
 - addressed 25
 - control information describing a record 17
 - keyed 23
 - mass sequential insertion 23
 - skipping 23

123

- 135 Central Processing Unit 14
- 145 Central Processing Unit 14
- 155 Central Processing Unit 14
- 165 Central Processing Unit 14
- 2305 Fixed Head Storage Models 1 and 2 14
- 2311 Disk Storage Drive 14
- 2314 Direct Access Storage Facility 14
- 2319 Disk Storage 14
- 3330 Disk Storage 14

**OS/VS Virtual Storage Access Method
(VSAM) Planning Guide
GC26-3799-1**

**Reader's
Comment
Form**

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name, job title, and business address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

fold

fold

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.



POSTAGE WILL BE PAID BY . . .

IBM Corporation
Programming Center—Publishing
Department D58
Monterey & Cottle Roads
San Jose, California 95193

fold

fold



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

OS/VS Virtual Storage Access Method (VSAM) Planning Guide Printed in USA GC26-3799-1



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)