IMS/VS - VSAM INTERFACE GUIDE

Document Number GG24-1518-00

Ulrich K. Schwenk

World Trade Systems Center
Santa Teresa
San Jose, California

This publication was produced using the IBM Document Composition Facility (program number 5748-XX9) Release 2. It was printed on the IBM 3800 Printing Subsystem.

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "As Is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Reference to PTF numbers that have not been released through the PUT process does not imply general availability. The purpose of including these reference numbers is to alert IBM CPU customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution rules.

The product referenced in this document may not be available in all countries.

First Edition (March 1980)

CONTENTS

1.0 INTRODUCTION

This document was created in response to the needs of new users of IMS/VS and those established users who are converting their data bases to VSAM. The purpose is to provide a guide and a reference for persons using VSAM as the operating system access method for their IMS/VS data bases.

1.1 OBJECTIVES OF THE DOCUMENT

The primary objective for this document is that it should act as a single, consolidated source of information on the interface between IMS/VS and VSAM as it affects the user of the system. Prior to this document, this information has been scattered through a large number of publications causing a great deal of time and effort to be spent in searching for the appropriate information. This document is not intended as a replacement for the SRL manuals and IBM courses, but as a supplement to them. It will be useful as an initial point of reference to give a basic understanding of specific areas, which may then be followed up by reference to the appropriate SRL manual or course.

A secondary objective of the document is to provide clarification of some of the areas of the interface between IMS/VS and VSAM which are prone to misunderstanding and may cause confusion to users. For this reason, there are many instances of redundancy in this document where information available from other sources has been repeated. This has been done to make this document more suitable as a learning and reference tool and not simply an index to other publications.

1.2 INTENDED AUDIENCE

This document has been written with the intention that it will be used by Data Base Administrators, System Programmers and Application Designers who are involved in implementing or maintaining an IMS/VS data base system using VSAM. The information contained herein is presented in such a way that it will be useful to these persons both as a guide or reference and as a means of learning about the relationship of IMS/VS and VSAM.

A certain level of familiarity with IMS/VS and VSAM has been assumed on the part of the reader as it is not the intention of this document to teach the basics of either IMS/VS or VSAM. The required level of knowledge may be attained by attending introductory courses on IMS/VS and VSAM or by completion of Independent Study Programs in these topics.

Although this document may be useful to them, the needs of IMS/360 and DOS DL/I users are not directly addressed.

1.3 SYSTEMS

The information in this document applies to OS/VS systems using the IMS/VS 1.1.5 system. The material is generally applicable to releases prior to IMS/VS 1.1.5. There are some instances when the implementation differs slightly on pre-IMS/VS 1.1.5 releases and no attempt has been made in this document to note these differences.

1.4 ACKNOWLEDGEMENTS

We want to acknowledge the debt we owe to the many individuals who gave us assistance with technical questions. In particular, we want to give credit to Rod Murchison from South Africa, whose significant participation in the development of this document is greatly appreciated, and also to Michel Quenon from the WTSC in Santa Teresa for his assistance in the IMS/VS and Fast Path area.

1.5 REFERENCE MATERIAL

Besides various internal sources the following publications have been used to develop this manual:

- GC26-3819-4 OS/VS VSAM OPTIONS FOR ADVANCED APPLICATIONS
- GC28-3838-3 OS/VS VSAM PROGRAMMER'S GUIDE
- G320-5774-0 VSAM PRIMER AND REFERENCE
- G320-6004-0 IMS/VS PERFORMANCE MONITORING & TUNING GUIDE
- G320-6015-0 OS/VS VSAM SHARING PART 1
- G320-6035-0 IMS/VS AND OS/VS BUFFER OPTIONS
- SH20-9025-6 IMS/VS VERSION 1 SYSTEM/APPLICATION DESIGN GUIDE
- SH20-9029-6 IMS/VS VERSION 1 UTILITIES REFERENCE MANUAL
- SH20-9145-0 IMS/VS VERSION 1 PRIMER

The following publications are referenced:

- GC26-3819-4 OS/VS VSAM OPTIONS FOR ADVANCED APPLICATIONS,
- GC26-3838-3 OS/VS VSAM PROGRAMMER'S GUIDE
- GC26-3840-3 OS/VS1 ACCESS METHOD SERVICES
- GC26-3841-2 OS/VS2 ACCESS METHOD SERVICES
- G320-5775-0 FAST PATH FEATURE DESCRIPTION AND DESIGN GUIDE
- SH20-9026-6 IMS/VS VERSION 1 APPLICATION PROGRAMMING REF. MANUAL
- SH20-9027-7 IMS/VS VERSION 1 SYSTEM PROGRAMMING REFERENCE MANUAL
- SH20-9028-6 IMS/VS VERSION 1 OPERATOR'S REFERENCE MANUAL
- SH20-9081-4 IMS/VS VERSION 1 INSTALLATION GUIDE

2.1 BASIC DESCRIPTION OF VSAM

Some topics of the following chapter are described in more detail in the document **VSAM PRIMER AND REFERENCE, G320-5774.**

2.1.1 Support by SCPs and Languages

Virtual Storage Access Method (VSAM) is a component of DOS/VS and OS/VS data management and is supported in DOS/VS, OS/VS1, OS/VS2 SVS, and OS/VS2 MVS.

In VM/370 the CMS/VSAM support is based on DOS/VS.

ASSEMBLER, COBOL/VS, PL/I, and RPGII (DOS/VS only) provide native VSAM support. They provide Sequential and Random (direct) accessing capabilities. For further discussions see the appropriate COBOL/VS, PL/I, or RPGII manuals.

2.1.2 RBA Concept in Addressing

The physical location of a logical record within a data set is given in the form of a **Relative Byte Address (RBA)** rather than a CCHHR disk record address.

The RBA of a logical record is the offset of this logical record from the beginning of the data set.

The first record in a data set has an RBA of 0. If the control interval contains more then one record, the second record has an RBA equal to the length of the first record and so on.

The RBA of a logical record or index entry, therefore, is independent of the physical characteristics of the direct access device type on which it resides, the number of extents in the data set, etc. It only depends on its position in the sequence of records. This way of referencing data by RBA makes the data sets independent of the DASD device type.

The RBA is always expressed as a full word binary integer.

The RBA includes free space and control information (Figure 3 on page 8 shows an example of how RBA values are assigned).

2.1.3 Control Interval and Control Area Definition and Concept

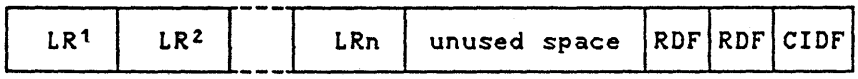
Control Intervals (CIs)

To make VSAM as efficient as possible, so-called **control intervals** have been designed to contain the logical records. The disk space on a direct access storage device (like IBM 3330, and IBM 3350) has been divided into logical blocks of information. The size of the blocks can vary from one VSAM data set to another VSAM data set, but for a specific data set, the size of each block is fixed, either by VSAM or by the the user (within limits acceptable to VSAM). These blocks of information are called **control intervals**. Figure 1 illustrates the logical records (and the control information describing them) stored in a control interval.

A control interval is the unit of data transferred between virtual storage and disk storage; for example VSAM will transfer the control interval shown in Figure 1 to virtual storage if one or more of the logical records in that control interval needs to be accessed.

The control interval must be a certain size (a multiple of 512 bytes unless the control interval size is greater than 8192, then the multiple is 2048). The maximum size of a control interval is 32,768 bytes. A control interval can span several tracks depending on control interval size and track capacity but it **cannot** cross control area boundaries (the control area is described on the next page).

A control interval always contains control information fields plus an integral number of logical records, and unused (free) space, or logical records alone, or free space alone. Figure 1 shows that control information fields consist of two types of fields: one CIDF (Control Interval Definition Field), and one or more RDFs (Record Definition Fields).



LR = logical record

Figure 1. Control interval structure with control fields

There is one **Control Interval Definition Field (CIDF)** per control interval and one **Record Definition Field (RDF)** per record. The RDF contains the length information for this record.

If records with the same length are used 2 RDFs are needed (IMS/VS uses VSAM records with the same length, even when variable length segments have been specified; only GSAM allows the use of VSAM variable length records). In this case one RDF contains the length information of a record and the other the number of records in this CI.

Only one RDF is used when a single record is stored in the control interval. Multiple RDFs will be used for GSAM variable length segments.

Control Areas (CAs)

The control intervals in a VSAM data set are grouped together into fixed length contiguous areas of direct access storage called **control areas**. A control area is auxiliary storage space that is set aside by VSAM to receive or hold the control intervals of a particular data set.

The user does not have to specify the number of control intervals per control area of a data set; VSAM determines this number. Whenever the space in a data set needs to be extended because the control area cannot hold anymore data, VSAM extends the data set (see also description of VSAM data spaces and space allocation in section "VSAM Data Space Allocation" on page 62).

The maximum size of a control area is **one cylinder** and the minimum size is one track of a DASD. The user implicitly defines the control area size by specifying the amount of space to be allocated to a data set (for further information, see section "Space Allocation (CYLINDER, TRACKS, RECORDS, VOLUMES)" on page 64). Usually better performance is obtained when the size of one control area is one cylinder.

Figure 2 shows a possible structure of two control areas, which include three control intervals each. All details, like 'free space', etc., are explained in following sections.

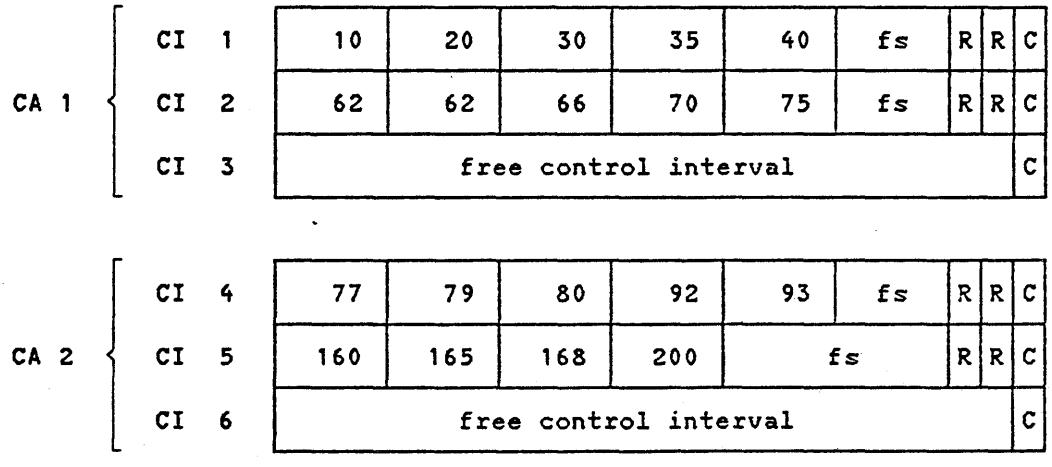


Figure 2. Structure of a VSAM control area

2.1.4 VSAM in Relation to Previous Access Methods

VSAM supports both sequential and direct processing and is designed to supersede ISAM, although the two access methods can coexist in the same Operating System. VSAM supports functions equivalent to those of ISAM, and provides better performance. VSAM data sets cannot be accessed by any other Access Method.

In addition, the three data organizations supported by VSAM can be used in place of BSAM or QSAM for sequential data sets and in place of BDAM for directly organized data sets. The new structure and features of VSAM make it more suited to data base and online environments than other access methods.

2.1.5 Suitability and Support in Data Base Environment

There are a number of facilities in VSAM which make it highly suitable to use in the data base environment. The most important of these are:

- Distributed free space which allows VSAM performance to be relatively insensitive to data set insert activity.
- The control interval concept which eliminates the need for ISAM 'overflow chaining' and keeps direct and sequential request times low.
- The structure and design of the VSAM index which reduces insert and direct retrieval time.
- The ease of control over the data sets gained through use of the Access Method Services utility.

These factors indicate that performance gains are likely to be achieved by migrating existing data bases from ISAM/OSAM to VSAM. The first-time IMS/VS user should also make use of VSAM to gain these advantages.

2.2 VSAM DATA SETS

VSAM supports three different data set organizations: **key-sequenced** (KSDS), **entry-sequenced** (ESDS), and **relative-record** (RRDS). All three organizations allow both sequential and direct processing and record addition without rewrite of the complete data set. The primary difference between these three organizations is the sequence in which logical records are stored.

2.2.1 Alternate Index (in brief - not used by IMS/VS)

VSAM alternate indexes enable the logical records of an ESDS or of a KSDS to be accessed sequentially and directly by more than one key. The alternate index itself is a VSAM KSDS.

None of the IMS/VS data base organizations make use of VSAM alternate indexes.

2.2.2 RRDS (in brief - not used by IMS/VS)

A **Relative Record Data Set** (RRDS) is processed by a relative record number and consists of a number of fixed length **slots**, each of which has a unique relative record number, an associated RDF, and contains one logical record. The slots within an RRDS are sequenced by ascending relative record number (from 1 to N).

A record is placed in the slot specified by a user-supplied relative record number (for direct or skip-sequential inserts) or, if not specified, by a VSAM-supplied relative record number (for initial load or sequential inserts). This relative record number is never included in the logical record.

None of the IMS/VS data base organizations make use of VSAM RRDS.

2.2.3 ESDS

An **Entry-Sequenced Data Set** (ESDS), which is logically comparable to a SAM data set, contains either records with the same length or variable length records, sequenced in the order in which they were submitted for inclusion in the data set (like SAM data sets). Therefore, records are sequenced by their time of arrival rather than by any field in the logical record.

An ESDS accepts records of the same length and variable length records (variable length VSAM records may be used in IMS/VS with GSAM only). Records can be added at the end, and records can be updated but not extended. Records cannot be erased.

Figure 3 on page 8 shows a possible structure of an ESDS.

The RBA values are in hexadecimal (4 bytes). The numbers in the CIs represent keys in the records (not used). The control interval length is 512 bytes. The small numbers below the CIs represent the record/field length in bytes.

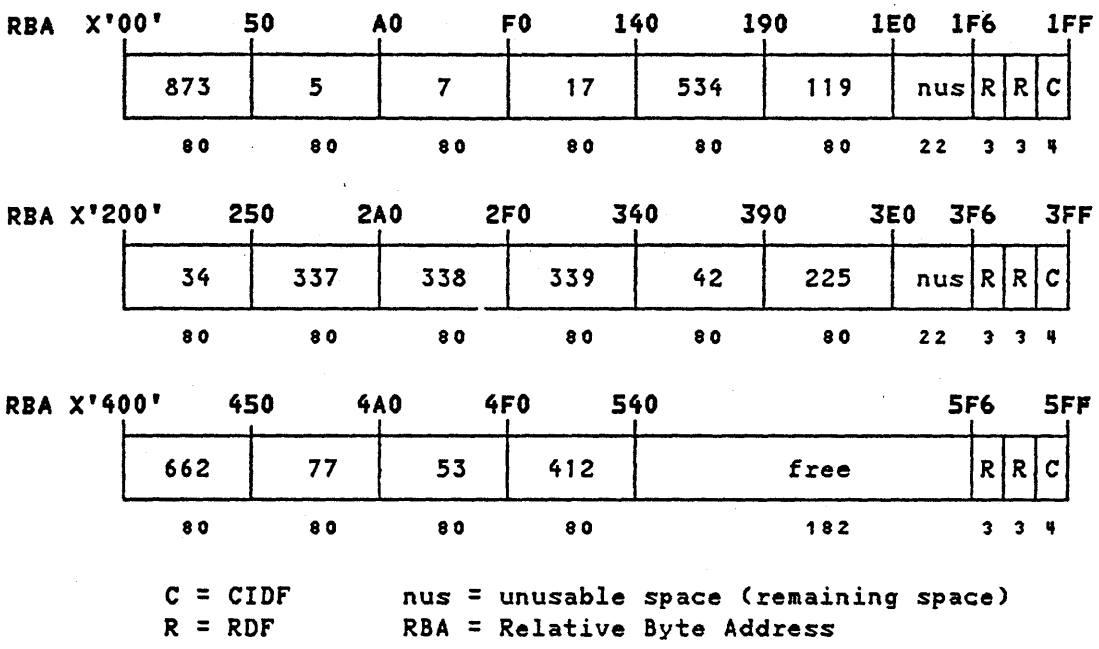


Figure 3. Entry-sequenced data set structure

2.2.3.1 Processing Restrictions

The following **request types** are allowed for ESDS processing:

- add/load records (only at the end)
- modify existing records (no record length change)
- retrieve records

Addressed sequential, addressed direct, and control interval processing are supported for ESDS.

The former two are called **addressed processing** and it is the normal way to process an ESDS.

No **keyed processing** is allowed, i.e. no key may be specified to access any record in an ESDS.

2.2.4 KSDS

A **Key-Sequenced Data Set (KSDS)** uses logical records, either with the same length or variable in length, which are placed in the data set in ascending collating sequence by a field, called the **key**.

Records added after the KSDS is created are inserted in key sequence and existing logical records are moved when necessary.

In a VSAM key-sequenced organization, a record must have a unique, embedded, fixed length key located in the same position within each logical record.

Keys can be a minimum of one byte and a maximum of 255 bytes.

2.2.4.1 Index Structure

A KSDS consists of an index (**index component**) and logical records (**data component**) with different data set (component) names. All extents of the KSDS data component must reside on the same type of direct access devices. The index component, however, can be placed on a device type different from that of the data component.

The index for a VSAM KSDS data set contains key values and pointers. It is built when the KSDS is initially loaded. A VSAM index also contains information about available space in the index component.

Although it is an internal VSAM function it should be mentioned that the key in an index entry is stored in a compressed form. This is called **key compression** (for detailed description of key compression see section "Key Compression" on page 43). For index calculations (see "Index CI Size" on page 36) an average compressed key length of 3 can generally be used.

In most cases a VSAM index consists of 2-3 index levels. The lowest level is called the **sequence set**. All levels above, which are automatically built by VSAM when necessary, are collectively referred to as the **index set**.

The **index set** consists of index records (CIs) which contain **index entries** (compressed) pointing to another index record on the next lower level.

In the sequence set, there is one index record per data CA. Each index record is designed to contain a number of index entries equal to the number of data CIs which fit in its CA.

An **index record** (equivalent to one index control interval) is fixed in length, either 512, 1024, 2048 or 4096 bytes (index CI sizes are described in chapter "Control Interval Sizes" on page 35).

Each index record contains a pointer to an index record in the next lower index level or to a control interval in the data component.

An **index entry** (in the index record) consists of the highest key associated with that CI (in compressed form), control information and a relative CI number, which will be multiplied by the CI size and added to the RBA of the beginning of the CA (derived from the index header). The sum addresses the correct data CI.

If a data set consists of more then one CA, more than one record (CI) exists in the index sequence set. In this case VSAM automatically builds another index level (L2). Each entry in the index L2 record points to one Sequence Set record (L1) (see Figure 4 on page 11).

If the data set is so big that the index L2 record cannot hold all entries for all the Sequence Set records, another index L2 record has to be built by VSAM.

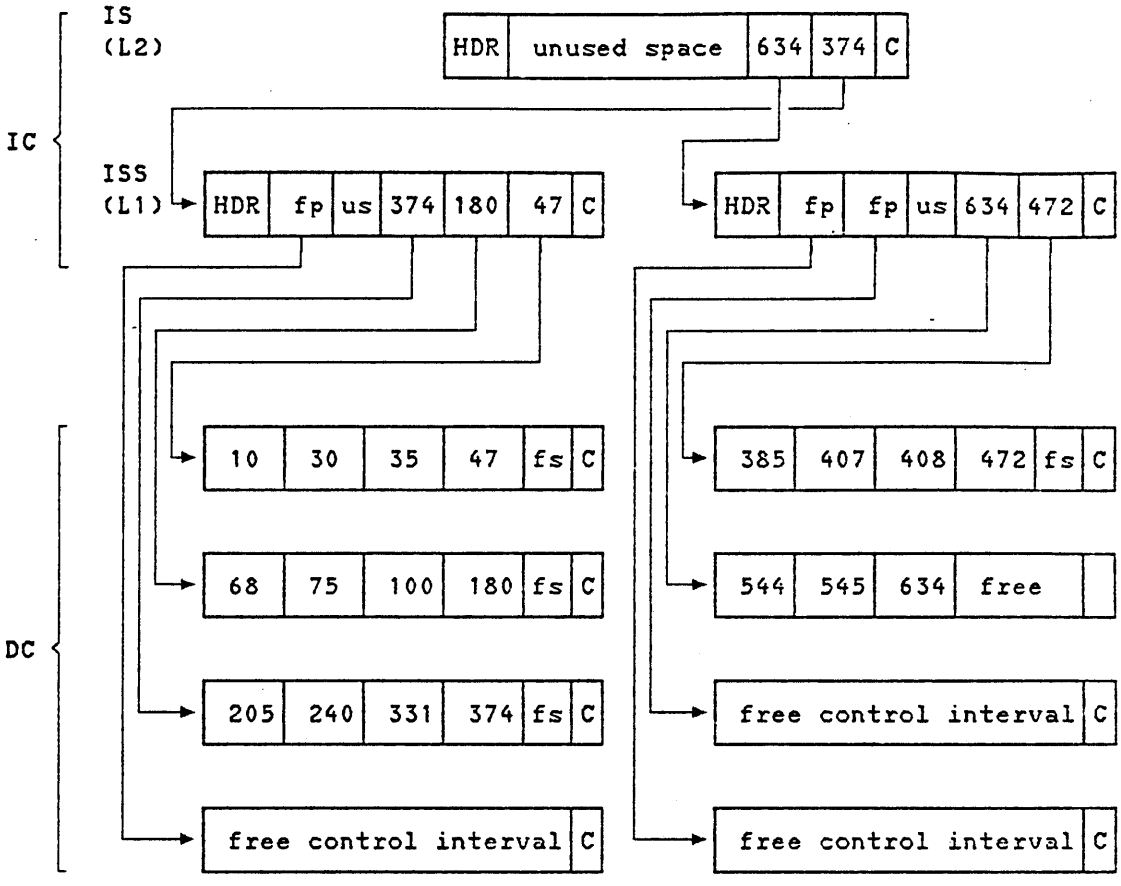
Since the highest index level can only consist of one index record, VSAM will build another index level (L3), etc.

For index performance considerations see section "Index CI Size" on page 36.

Figure 4 on page 11 shows a logical picture of a KSDS after loading with two CAs, containing four CIs each.

Records with 80 bytes each were loaded. A free space value of 20% free space in the CI and 25% in the CA was specified (see "FREESPACE" on page 73), so 1 record (minimum = (512 * 20%) = 102) per CI and one CI per CA (25% * 4) (CI/CA) is held free when loading the data set. In each CI with a length of 512 bytes and control fields with a total of 10 bytes, 4 logical records were loaded (102 bytes were left free).

The index component consists of two levels: the Index Sequence Set and the Index Set.



C = control fields (RDFs, CIDF)
DC = Data Component
fp = pointer to free CI
fs = free space (FREESPACE def.)
HDR = Index header information
IC = Index Component (IS + ISS)
IS = Index Set
ISS = Index Sequence Set
L1,L2 = Index Level 1 and 2
us = unused space

Figure 4. Key-sequenced data set structure

2.2.4.2 Free Space

A KSDS can be defined to reserve space for later insertions when the data set is loaded or updated using a sequential insertion technique (e.g. PUT SEQ). This reserved space is called **free space**.

The amount of free space in a CI and/or CA (in %) is specified in the FREESPACE parameter of the DEFINE CLUSTER command (see also "FREESPACE" on page 73).

Use of VSAM Free Space for Insert

When a record must be added to a control interval, records with keys higher than the one to be inserted are shifted to the right of the CI within the free space limit. As long as enough free space is available in this CI, there is no further data CI manipulation.

Sequential Access is used for loading data sets and for insertion of groups of pre-sorted records and is also called 'Mass Insertion'. Mass insertion is described in detail in section "Sequential Processing" on page 98.

Skip-sequential Access is described in section "Skip-sequential Insertion" on page 99.

The following examples illustrate the usage of free space:

This CI is 512 bytes long. A free space value of 30% has been specified. With a record length of 80 bytes, the maximum CI capacity is 6 records. But after loading the data set, each CI contains only 4 records and 182 bytes of free space (VSAM calculated a minimum of 153 bytes free space).

10	20	30	40	free space	R	R	C

Record 25 is to be inserted. The control interval is read into a VSAM buffer and records 30 and 40 are shifted to the right.

10	20	25	30	40	fr.sp.	*R	R	* C

Records 20 and 40 are to be deleted. The remaining records are shifted to the left if necessary and the freed space is now available for inserts, updates, etc.

10	25	30	free space	*R	R	* C

R = RDFs (Record Definition Fields)
C = CIDF (describes free space and offset)
* = value changed

Note: Not one of these changes require a change in the related index sequence set, even if the high key record in a CI is deleted.

Figure 5. Inserting and deleting VSAM records

If the record to be inserted or to be expanded will not fit in the control interval, a **control interval split** takes place to provide sufficient space in the control interval to contain the record. The control interval split is described in detail in section "Splitting and Insert Strategies" on page 29.

2.3 DATA BASE ORGANIZATIONS SUPPORTING VSAM

2.3.1 HSAM, SHSAM - no VSAM support

There is no support for VSAM in the Hierarchical Sequential Access Method (HSAM). All HSAM or SHSAM (Simple HSAM) data bases are implemented using either BSAM or QSAM data sets. In IMS/VS DB/DC (online), only BSAM data sets may be used for HSAM data bases.

2.3.2 HISAM, SHISAM

Data bases in the Hierarchical Indexed Sequential Access Method (HISAM) may be implemented using either ISAM/OSAM or VSAM data sets. An HISAM data base will consist of two data sets, one used for primary storage of the data base records and the other used for overflow storage. In the case of a SHISAM (Simple HISAM) data base which contains only one segment type, just one data set is used.

Logical records in the data sets are of same length, even when variable length segments are defined in the data base. The user will define the logical record size and the block or control interval size for each data set.

A data base record may span multiple logical records in the data sets. The minimum storage requirement for each data base record is a single logical record in the primary data set. This will be an ISAM record when the data base is using ISAM/OSAM or a KSDS record when VSAM is used. OSAM or ESDS records from the overflow data set are used when a data base record becomes larger than a single logical record.

The number of logical records spanned depends upon the size and frequency of occurrence of the segments within the data base record. A pointer chain is used to link all of the logical records in both the primary and overflow data sets which are used for storage of the data base record. Within the logical records which make up the data base record, the segments are kept in hierarchical sequence and are related by physical adjacency. Figure 6 on page 14 and Figure 7 on page 14 show the way in which data base records are stored in an HISAM data base with ISAM/OSAM and VSAM.

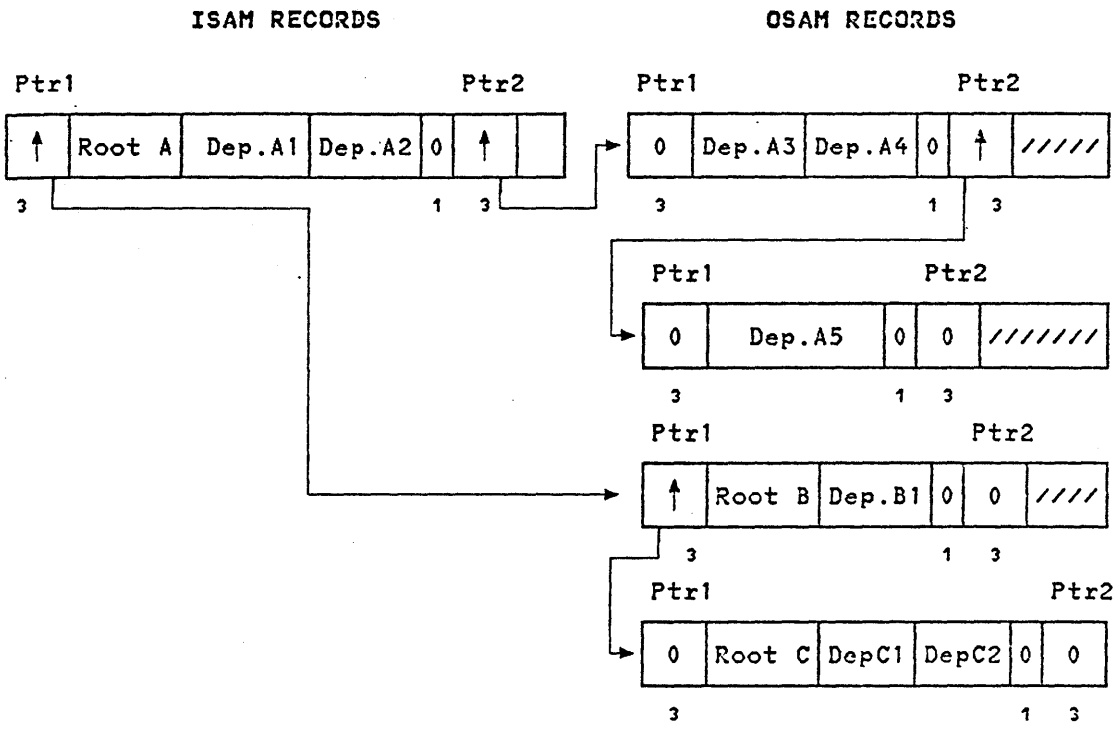


Figure 6. HISAM data base records in ISAM/OSAM: 'Ptr1' is used for the Root Overflow Chain and 'Ptr2' is used for the Dependent Overflow Chain.

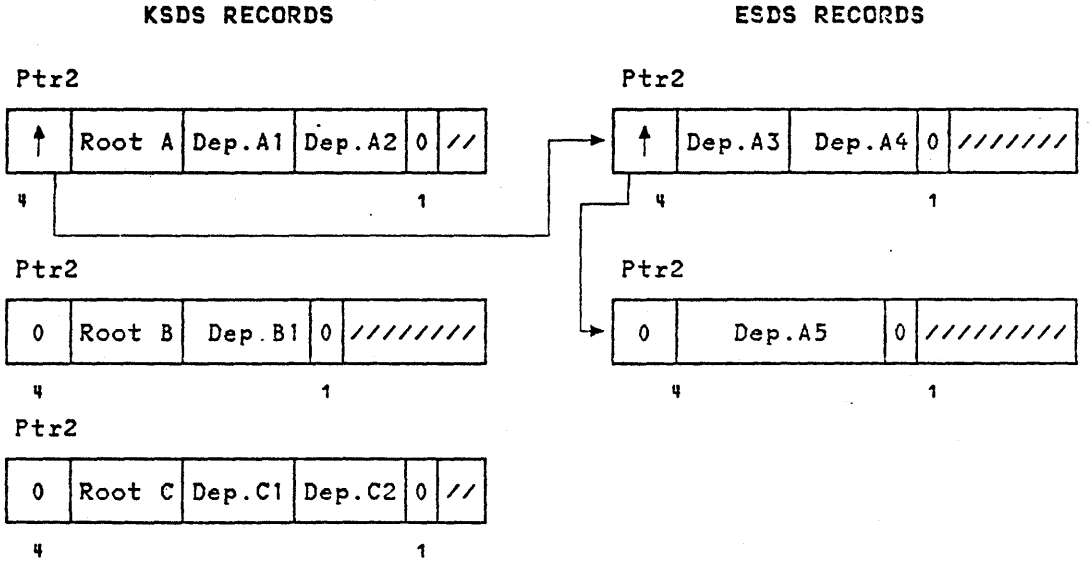


Figure 7. HISAM data base records in VSAM: 'Ptr2' is used for the Dependent Overflow Chain.

In an HISAM data base, the data base records are maintained in root key sequence. As explained below, the maintenance of the root key sequence will be done using different methods in ISAM/OSAM and VSAM. The Dependent

Overflow Chain, 'Ptr2' in Figure 6 and Figure 7, is used to link all of the logical records which comprise the data base record and allow access to all of the segments of that data base record in hierarchical sequence.

When ISAM/OSAM is used, roots inserted after initial load ('Root B' and 'Root C' in Figure 6 on page 14) are placed in new OSAM logical records in the Overflow data set. The **Root Overflow Chain**, 'Ptr1', is used to maintain the root segments in root key sequence.

When the data bases use VSAM, new roots which are inserted after initial load ('Root B' and 'Root C' in Figure 7 on page 14) are placed in new KSDS logical records in the Primary data set. This direct insertion into the Primary data set is possible because of the structure and design of the VSAM index. No Root Overflow Chain is necessary with the VSAM implementation and this saves on the amount of system overhead required in each logical record.

A Simple HISAM (SHISAM) data base is only allowed with VSAM implementation. The SHISAM data base may contain only root segments. Since no dependent segments exist, no Dependent Overflow Chain need be maintained nor is the ESDS Overflow data set required. A SHISAM data base, therefore, is totally equivalent to a VSAM KSDS.

2.3.3 Direct Organizations

In the direct organizations , Hierarchical Direct Access Method (HDAM) and Hierarchical Indexed Direct Access Method (HIDAM), the data base records are stored in a different format from that used in HISAM. For HISAM, each data base record has its own individual set of logical records linked together by the Dependent Overflow Chain and the segments are stored in these records in hierarchical sequence thereby being related through physical adjacency. The direct organizations do not assign a logical record to a particular data base record but, will place segments from different data base records into the same logical record. Furthermore, the segments in direct organizations are related through pointers associated with each segment rather than by physical adjacency.

These pointers are stored as part of each segment in the data base. As shown in Figure 8 on page 16, each segment consists of two portions:

- The prefix, containing a 1-byte segment code, a 1-byte delete flag and a number of 4-byte pointers, and
- The segment data which is the information actually passed to application programs in response to a request for this segment.

In an ISAM/OSAM data base these pointers are direct address type pointers. In a VSAM data base, the pointers are Relative Byte Addresses (RBAs). For complete information on the pointers that may be used in HDAM and HIDAM, refer to the manuals **IMS/VS VERSION 1 SYSTEM/APPLICATION DESIGN GUIDE, SH20-9025** and **IMS/VS VERSION 1 UTILITIES REFERENCE MANUAL, SH20-9029**.

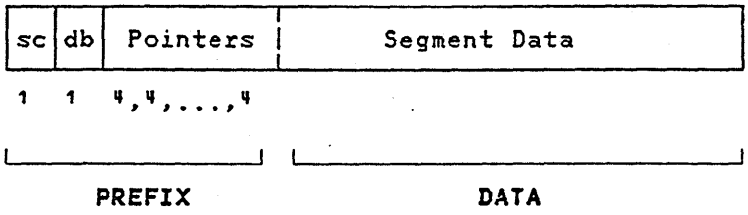


Figure 8. Format of a segment in an HDAM or HIDAM data base.

2.3.3.1 HDAM

An HDAM data base consists of only one data set, either an OSAM data set or a VSAM ESDS. This data set is divided into two portions, the **Root Addressable Area** where root segments and dependents are stored and the **Overflow Area** where root segments and dependents that do not fit into the Root Addressable Area are inserted after initial load. The sizes of these areas are defined by the user in the RMNAME parameter of the DATASET macro as explained in "RMNAME" on page 51.

Access to root segments in the HDAM data base is achieved through the use of a Randomizing Routine. This randomizing routine takes a root key as input and returns a relative block number for OSAM or a control interval number for VSAM and a Root Anchor Point (RAP) number.

The number of RAPs in each block or control interval is also defined in the RMNAME parameter. A root segment is located by accessing the indicated block or control interval and following the pointer or chain of pointers from the specified RAP to the desired root segment.

Figure 9 on page 17 shows the pointers used in an HDAM data base to link from RAPs to root segments and to relate the segments. There is no real difference except for the use of blocks or control intervals between the implementation of an HDAM data base with OSAM or VSAM.

In each block or control interval there is one **Free Space Anchor Point** which points to the first Free Space Element in the block or control interval. The **Free Space Elements** are used to track and control the unused areas in the block or control interval and are referenced whenever a segment is placed into or deleted from the block or control interval.

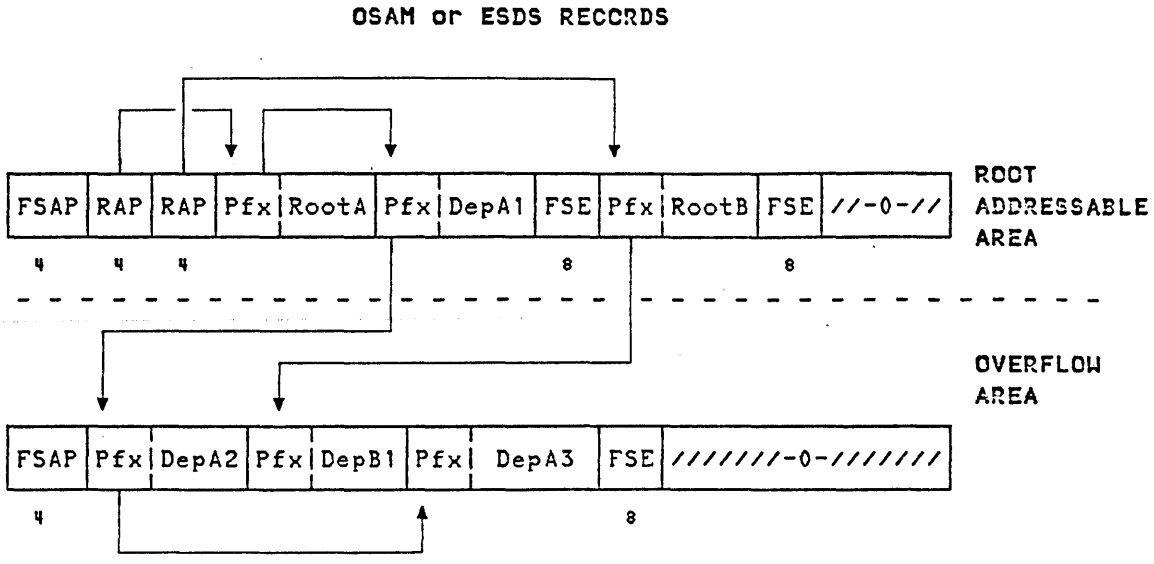


Figure 9. HDAM data base records in either ISAM/OSAM or VSAM.

2.3.3.2 HIDAM

An HIDAM data base has two components. The first is the **Data Portion** which is very similar to an HDAM data base and where all of the segments in the data base are stored. The other component is the **Primary Index** which is a separate data base which is structured like an HISAM data base. HIDAM data bases may therefore be regarded as a combination of HDAM and HISAM.

The segments of the Primary Index of an HIDAM data base are not completely like HISAM segments in that they each have a prefix. This prefix contains a delete flag and a single pointer to a root segment in the Data Portion of the data base. The segment data in the Primary Index consists of the root key and the index segments are maintained in root key sequence. Figure 10 shows the format of an index segment in the Primary Index of an HIDAM data base.

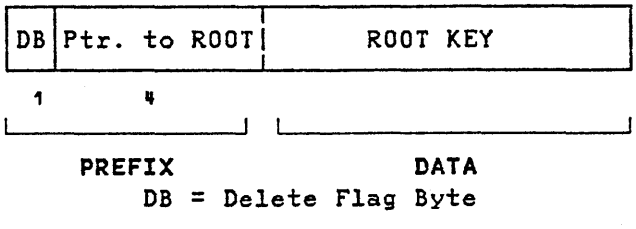


Figure 10. Format of a Segment in an HIDAM Primary Index

Access to HIDAM root segments is performed by retrieval of the appropriate Primary Index segment and following the prefix pointer to the corresponding root segment. Dependent segments are accessed through the pointers in the prefixes of the segments in the Data Portion of the data base just as with HDAM.

Figure 11 and Figure 12 on page 19 show the implementations of an HIDAM data base using ISAM/OSAM and VSAM. To simplify the figures, Free Space Anchor Points, Free Space Elements and Root Anchor Points have not been shown. The use of FSAPs and RAPs in HIDAM is covered in section "BLOCK, RECORD and SIZE" on page 53 and shown in Figure 34 on page 57.

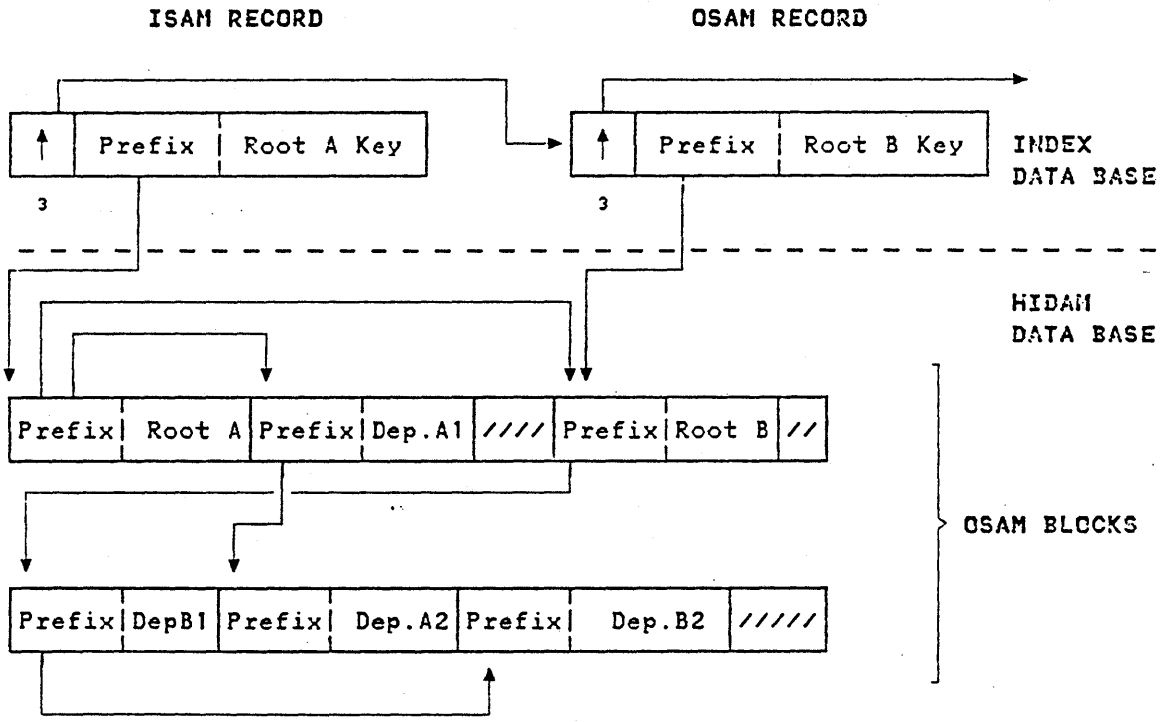


Figure 11. HIDAM data base records in ISAM/OSAM: This figure shows the relationship between the data base records and the Primary Index of the HIDAM data base.

It should be noted that two data sets, one ISAM and one OSAM, are required for the Primary Index of an HIDAM data base implemented with ISAM/OSAM.

The reason that two data sets are used for the Primary Index is that, when a new root segment is inserted into the data base after initial load ('Root B' in Figure 11), the corresponding Index segment is placed in the OSAM data set and the Root Overflow Chain pointers are used to maintain the root key sequence.

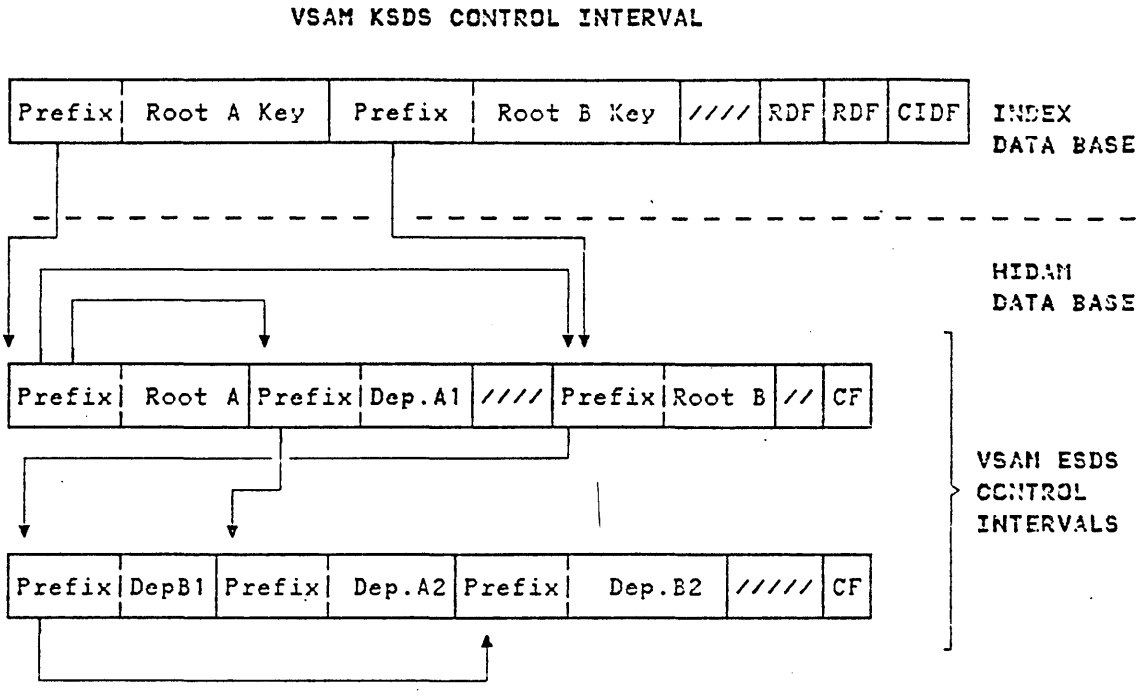


Figure 12. HIDAM data base records in VSAM: This figure is intended to show the difference in the Index data base for an HIDAM data base using VSAM as opposed to ISAM/OSAM.

The Primary Index of an HIDAM data base consists of a single KSDS when the data base is implemented using VSAM. Unlike the case with ISAM/OSAM, VSAM's index structure allows the Index segment for a new root to be inserted directly into the data set in its proper position to maintain root key sequencing.

2.3.4 INDEX Data Bases

An **INDEX** data base is a root-only HISAM data base used as a means of accessing segments in some other data base by key. There are two types of INDEX data base, the HIDAM Primary Index and the **Secondary Index**.

The HIDAM Primary Index may be implemented with either ISAM/OSAM data sets or with a VSAM KSDS as described in the preceding section "HIDAM" on page 17 and shown in Figure 11 on page 18 and Figure 12. A Secondary Index may only be created using VSAM data sets.

The structure of the Secondary Index data base will depend upon whether or not unique keys are used.

If unique keys are used, then the INDEX data base is treated as a Root-only HISAM data base and is implemented as a single KSDS.

For the nonunique key case, multiple occurrences of the same key value will be treated in a manner similar to that used for overflow dependents and the INDEX data base will be implemented as two VSAM data sets, one KSDS and one ESDS.

Figure 13 shows the structure of a Secondary Index with unique keys and Figure 14 on page 21 shows the structure for nonunique keys.

A Secondary Index may be defined on an HISAM, HDAM or HIDAM data base. Although the Secondary Index data base must be VSAM-based, there is no such restriction on the data base being indexed which may be ISAM/OSAM or VSAM.

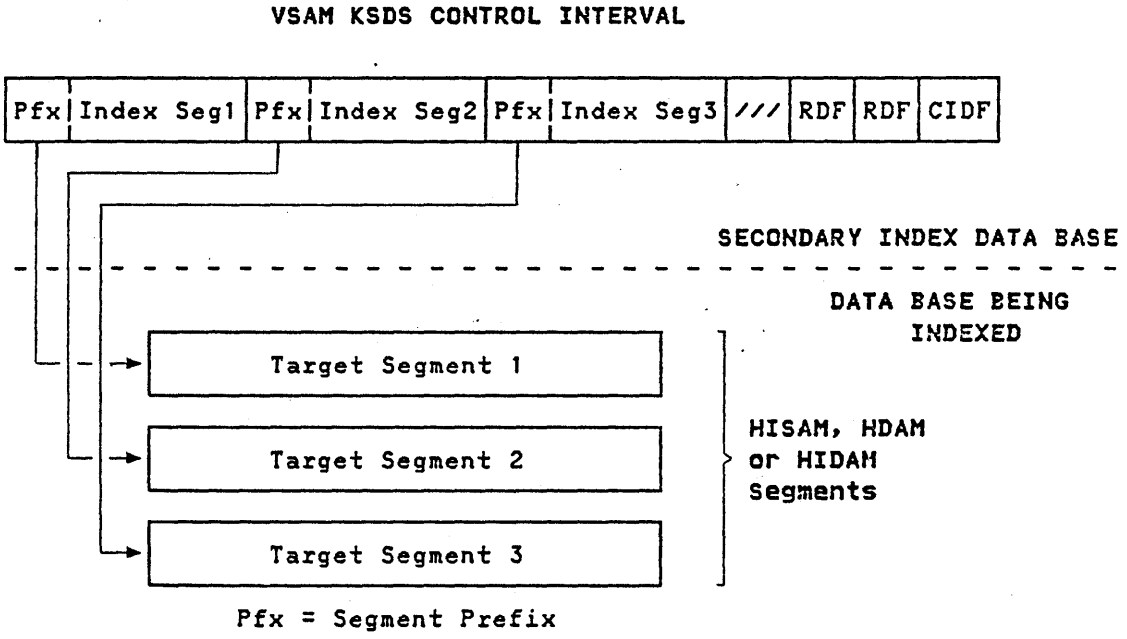


Figure 13. Unique Key Secondary Index

When the Secondary Index has unique keys, the INDEX data base is a normal VSAM KSDS. The index segments are maintained in secondary key sequence using the VSAM index structure.

The type of pointer used in the prefix of the index segments depends on the type of data base being indexed.

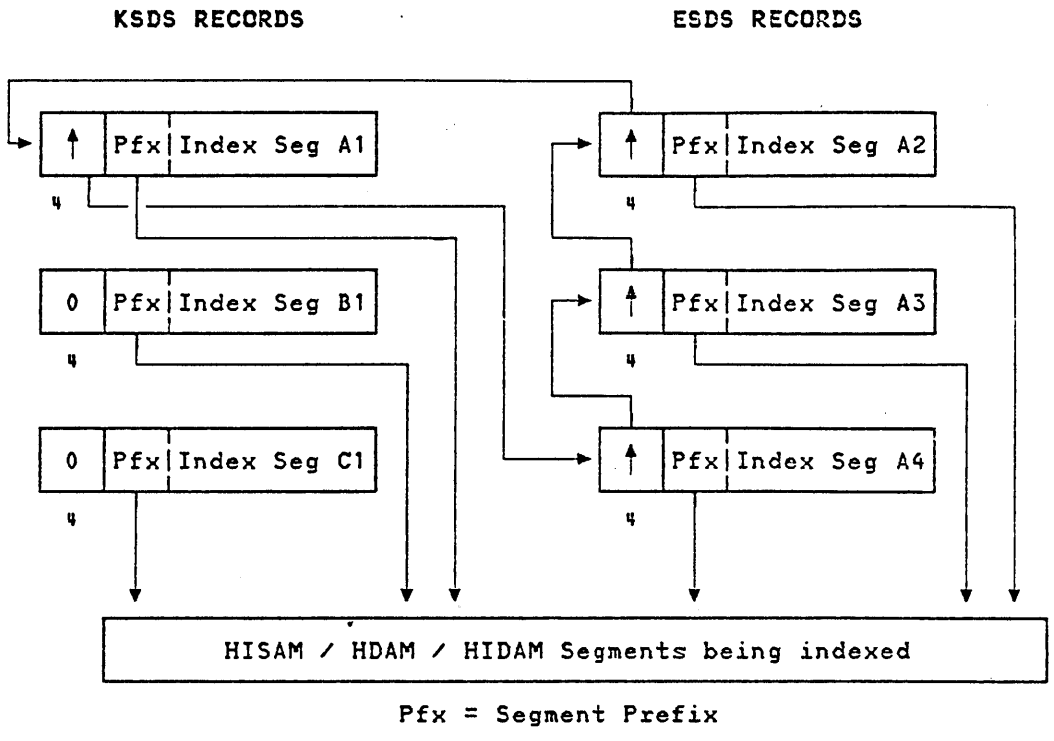


Figure 14. Nonunique Key Secondary Index

When the Secondary Index has nonunique keys, the Root Overflow Chain is used to link together all of the index segments with the same key. A second data set, an ESDS, is required to store those index segments whose keys are the same as an existing index segment.

It should be noted that the duplicate key index segments are maintained in Last-In-First-Out sequence on the Root Overflow Chain. This means that, whereas the 'A' index segments in Figure 14 were inserted in the order: A1, A2, A3, A4; they will be retrieved in the order: A4, A3, A2, A1.

The VSAM index structure is still used to maintain the index segments which have different keys in secondary key sequence.

2.3.5 Data Entry Data Bases

Data Entry Data Bases (DEDB), used with the **FAST PATH FEATURE**, are implemented as a group of VSAM Entry Sequence Data Sets. Only VSAM is allowed as the access method for a DEDB.

A full description of the Fast Path Feature and DEDBs may be found in "Fast Path Feature" on page 103.

2.3.6 GSAM

The Generalized Sequential Access Method (GSAM) provides accessing support for simple physical sequential data sets, such as tape files, SYSIN, SYSOUT, and others that are not hierarchical. These are data sets which, before GSAM, could not be used as IMS/VS data sets.

Support provided includes sequential or direct retrieval by a record identifier which defines the relative position of that record.

Support is provided for both OS/VS Sequential Access Method (SAM) and OS/VS Virtual Storage Access Method (VSAM) for Entry Sequenced Data Sets (ESDS).

GSAM is fully described in **IMS/VS APPLICATION PROGRAMMING REFERENCE MANUAL, SH20-9026.**

3.1 VSAM FREE SPACE AND DL/I FREE SPACE

The ability to define free space in the data sets used as data bases can materially improve the performance of the system. Some of the advantages to be obtained through the use of free space are:

- The overhead associated with insertion becomes more nearly that of update because space for new segments has been effectively 'reserved' by free space.

- There is no need to extend the data set frequently in order to provide space for new segments.

- Retrieval of an inserted segment is as efficient as retrieval of a segment created at initial load because free space allows the physical storage sequence of the segments to closely correspond to the logical sequence.

- The need for lengthy and complex procedures to obtain space for a new segment is avoided.

Although free space can be a means of improving performance, it is also possible to make a poor choice for the amount of free space which will have an adverse effect on performance. Therefore, the purpose of this section is to enable the readers to gain a sufficient understanding of the free space concept to make the proper choice for their own environment.

3.1.1 Applicability of Free Space

There are two separate and distinct forms of free space available when VSAM data sets are used for IMS/VS data bases. One is the Native VSAM free space which is specified when the VSAM cluster is defined and the other is DL/I free space which is specified as part of Data Base Description (DBD) generation.

3.1.1.1 VSAM - KSDS Free Space

Native VSAM free space applies only to KSDS and may not be specified for either ESDS or RRDS.

Two values may be specified for the VSAM free space:

control interval free space is specified as a percentage of each control interval to be left free for sequential inserts.

control area free space is specified as a percentage of the number of control intervals in each control area to be left free for sequential inserts.

VSAM free space is specified using the **FREESPACE** parameter as explained in section "FREESPACE" on page 73.

The primary function of VSAM free space is to allow insertions without the need for control interval or control area splits. Control interval and control area splits are explained in the following section "Splitting and Insert Strategies" on page 29.

3.1.1.2 DL/I - ESDS Free Space

Unlike VSAM free space, DL/I free space applies to the data component of HDAM and HIDAM data bases which are Entry Sequence Data Sets when the data bases are implemented with VSAM. Thus, one obtains free space in a KSDS with VSAM free space and in an ESDS with DL/I free space.

DL/I free space is requested in the **FRSPC** parameter of the **DATASET** macro for DBD generation. The FRSPC parameter is specified as:

FRSPC=(free block frequency factor, free space percentage factor).

The **'free block frequency factor'** (FBFF) may be specified as '0', the default, or as an integer from '2' to '100', inclusive. It is not permitted to specify '1' as the value of the FBFF. When a nonzero value, 'n', is given, then every 'n'th control interval in the ESDS will be left as free space during data base load or reorganization. If '0' is specified or allowed to default, then no control intervals will be left free.

The permissible values for **'free space percentage factor'** (FSPF) are the integers from '0' to '99'; '0' is the default value. FSPF specifies the minimum percentage of each control interval in the ESDS that is to be left as free space during data base load or reorganization. There is a case when less than the specified minimum free space may be allowed during load or reorganization. This will occur when a segment whose size is such that the total of the segment size and the free space specification exceeds the size of the control interval is to be loaded. When loading these 'oversized' segments, the FSPF is ignored and they are each placed into a control interval on their own.

The primary purpose for the use of DL/I free space is to facilitate the space search process used when inserting segments in HDAM or HIDAM data bases. This space search process attempts to find space to insert a segment either in the same control interval as the segment preceding it in hierarchical sequence or in a control interval close to one containing the hierarchically previous segment. If no space in the same control interval or in a nearby control interval is available, then the space search process begins a somewhat lengthy and complex procedure in order to find the location at which to insert the new segment. DL/I free space uses the FSPF to reserve space in the control intervals for segment insertion and the FBFF to ensure that there will be a nearby control interval with space available.

3.1.2 Control Interval Free Space and FSPF

Native VSAM control interval free space and DL/I free space percentage are very similar in their function in that they are both intended to allow inserts with minimal overhead. As long as there is available free space in the control intervals, new segments may be created without any need for the additional processing of control interval splits or extended space search. Insertion into free space also avoids lengthy retrieval paths as the new segments will be placed in logical sequence so that they may be located more rapidly.

These two forms of free space may therefore be regarded as the first line of defence against degradation of performance due to high levels of insert activity. As a result, a certain amount of care and attention should be given to the use and specification of this level of free space.

3.1.3 Control Area Free Space and FBFF

The VSAM control area free space and DL/I free block frequency are not as crucial to performance, but are still important. Their functions are not as closely related as are those of the two other forms of free space. Control area free space is intended to prevent the need for control area splitting, which in some cases may involve the movement of an half cylinder of data across a large distance, when no more free control intervals are available for splitting. The purpose of DL/I free blocks, on the other hand, is simply to leave space near the control interval in which it is initially desired to place the new segment so that lengthy space search processing may be avoided.

As these forms of free space are only required when the free space within loaded control intervals is insufficient or unavailable, a certain degree of 'guesswork' is acceptable in their specification. However, this level of free space should not be ignored as its omission may cause sudden, severe degradation of performance when the free space in the control intervals is exhausted.

3.1.4 Guidelines for Use of Free Space

Since each installation will have different data base structures and processing profiles, it is not possible to give specific rules and algorithms for the determination of the amount of free space which should be allocated. Therefore, this section is merely intended to make the reader aware of some of the considerations to be taken into account when specifying free space.

3.1.4.1 Free Space in KSDS

KSDS free space is applicable to SHISAM data bases, the Primary component of HISAM data bases and to INDEX data bases. In the case of an INDEX data base which is used as a Secondary Index and which has nonunique keys, it applies only to the KSDS portion of the data base. The use of VSAM free space with these types of data base can be an important factor in performance.

When defining the amount of CI and CA free space for a KSDS, the user should take into account the fact that too little free space may cause a large number of control interval and control area splits. This will result in poor performance because:

- CI/CA splits are time consuming since they involve the movement of data between control intervals and cause index modification.
- After a split, the records are no longer physically in sequence which will cause extra time to be spent in sequential processing.
- CA splits will increase seek time for direct requests because new CAs are placed at the end of the data set.

In order to avoid these problems, the user should specify sufficient free space to allow for the predicted insert level for the data base. It will usually be possible to make a fairly good estimate of the insert activity to be expected in a data base but, if not, then careful monitoring of the data base in the operational environment will provide the necessary information on which to base the calculation of free space percentages.

When allocating free space, it is also necessary to consider the processing mode of the data base. VSAM free space performs its function best for direct insert requests. If the data base is processed in sequential mode with mass sequential inserts being done, then free space may be relatively useless or even harmful. This is because mass sequential insertion preserves the free space percentages as specified in the definition of the cluster. Contiguous records are loaded into a control interval until the free space threshold is reached, then the next control interval will be used (no split). This may lead to unwanted free space between contiguous records.

There are also some other problems that may result from an overly generous allocation of free space. These problems are primarily due to the fact that the records will be spread over a larger amount of DASD space when imbedded free space is included. This may cause:

- Increased seek time because of the larger number of cylinders occupied by the data set.

- Slower direct retrieval because of extra index levels resulting from the fact that more index records are needed to reference the same amount of data when it is spread over a larger number of control areas.

From the above discussion, it can be seen that selection of the VSAM free space percentages is a matter of balance. Too little and too much free space can both have adverse effects on the performance of the data base. It is, however, possible to monitor the performance of the data bases with both IMS/VS utilities and the Access Method Services **LISTCAT** command (examples are shown in "Statistical Information" on page 125). If a large number of CI and CA splits are occurring, then the free space should be increased. Should it be found that frequent mass sequential inserts are being performed or that direct processing is slowed due to extra seek time or index search, then free space should be decreased.

The free space percentages for VSAM data sets can be easily changed by making use of the **ALTER** command of Access Method Services (the newly specified free space value has no effect for the existing data in the data set but will be used for future sequential insertions). For further information on ALTER, refer to **OS/VS1 ACCESS METHOD SERVICES, GC26-3840** or **OS/VS2 ACCESS METHOD SERVICES, GC26-3841.**

VSAM free space definition (for KSDS) is described in section "FREESPACE" on page 73.

3.1.4.2 Free Space in ESDS

Free space in an ESDS applies only to HDAM data bases and to the Data component of an HIDAM data base. Although other IMS/VS data base organizations use ESDS, only the Hierarchical Direct organizations may be defined with the **FRSPC** parameter to specify DL/I free space.

The basic purpose of free space in HDAM and HIDAM is to allow the space search process to find space for a new segment close to the segment preceding it in hierarchical sequence. This is intended to keep the physical storage of the data base in as close to logical sequence as possible. Thus, DL/I free space should be specified so that it can handle the expected number of insertions to the data base. The expected insertion level is usually easily determined by a simple analysis of the applications running against the data base.

If insufficient DL/I free space is allocated, then the following problems may be expected:

- Long insert times due to the space search process having to spend a great deal of time to find the proper location for new segments.

- Poor retrieval performance because of the fragmentation of data base records across the DASD space, resulting in a large amount of seeking.

- Very slow sequential processing of the data base because of the fragmentation of data base records and loss of logical sequence.

There are also problem areas that may be encountered when too much free space has been specified. These problems are more severe in HDAM than in HIDAM because of the use of the randomizing routine rather than an Index.

For an HIDAM data base, the segments are initially loaded in hierarchical sequence and the Index points to the location of each root segment. The HDAM randomizing routine returns a control interval number and a Root Anchor Point (RAP) number and it is expected that the root segment should be placed in that control interval and pointed to by the indicated RAP. If there is not enough space to insert a root segment into the specified CI, then it is placed elsewhere, as determined by the space search process, and the RAP is set to point to the actual location of the segment.

When a small value is specified for FBFF, requesting a high frequency of free control intervals, it makes it much more likely that root keys will randomize to a CI which must be left free. Therefore, they must be placed in another location. If FSPF is high, then fewer segments can be loaded into each control interval leading to the excess segments being spilled into some other control interval. These displaced segments will take up space in other CIs and may prevent segments which should be placed in those CIs from being inserted in their proper location so that they too are displaced. This displacement will cause other 'collisions' with the result that, in extreme cases, very few segments will be located in the CI to which their keys have randomized and the data base records will be badly scattered. Such an effect leads to lengthy insert processing and poor retrieval times.

The IMS/VS utilities may be used to determine if this situation has arisen by reporting the frequency with which segments are found out of their 'home' (randomized) CIs. In this case, it may be possible to reduce the problem by cutting down on the amount of free space specified for the data base and reloading it.

DL/I free space (for ESDS) specification is described in section "FRSPC" on page 58.

3.2 SPLITTING AND INSERT STRATEGIES

Splitting of control intervals and control areas can only occur in a KSDS. Therefore, splitting considerations apply only to HISAM, SHISAM, and INDEX data bases.

3.2.1 Control Interval Splitting (Direct Insert)

If the record to be inserted or to be extended will not fit in the control interval, a **control interval split** takes place as illustrated in the following example.

- Assume the following control area contains 3 control intervals.

  A free space value has been specified to reserve 33% of the number of CIs to be used for later insertions. This results in one free control interval per CA (33% of 3 control intervals).

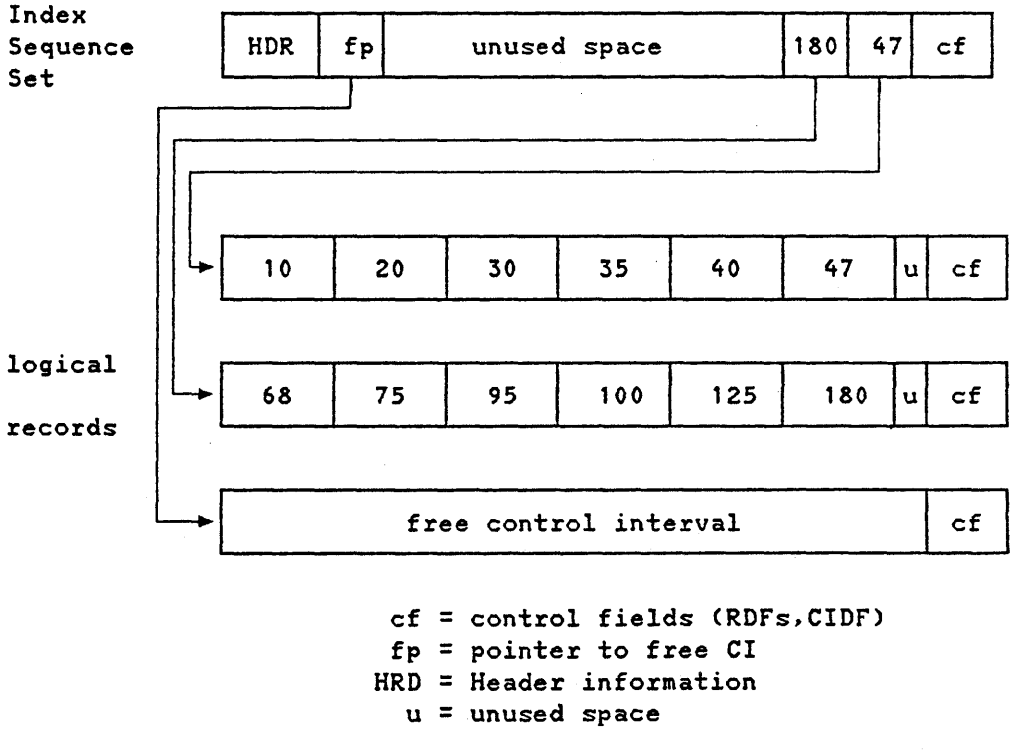


Figure 15. Control interval split (1 of 2)

- Now record 25 is to be inserted. In this example, the first data control interval is read by VSAM into a VSAM buffer. By examining the control fields VSAM determines that there is not enough free space available for this record.

  About half of the control interval is moved from this buffer to a second empty buffer (in OS/VS systems it is already reserved for control interval split by VSAM). Then the new record is inserted into the appropriate buffer. If, as in this example, there is an empty control interval available (at the end of the control area), the second buffer is written into this empty control interval on auxiliary storage.

  After this, the physical sequence of control intervals within the CA no longer represents the correct sequence of the logical records. Therefore, the primary index is updated to reflect this condition.

  Finally, the first buffer is written back into the old control interval on auxiliary storage.

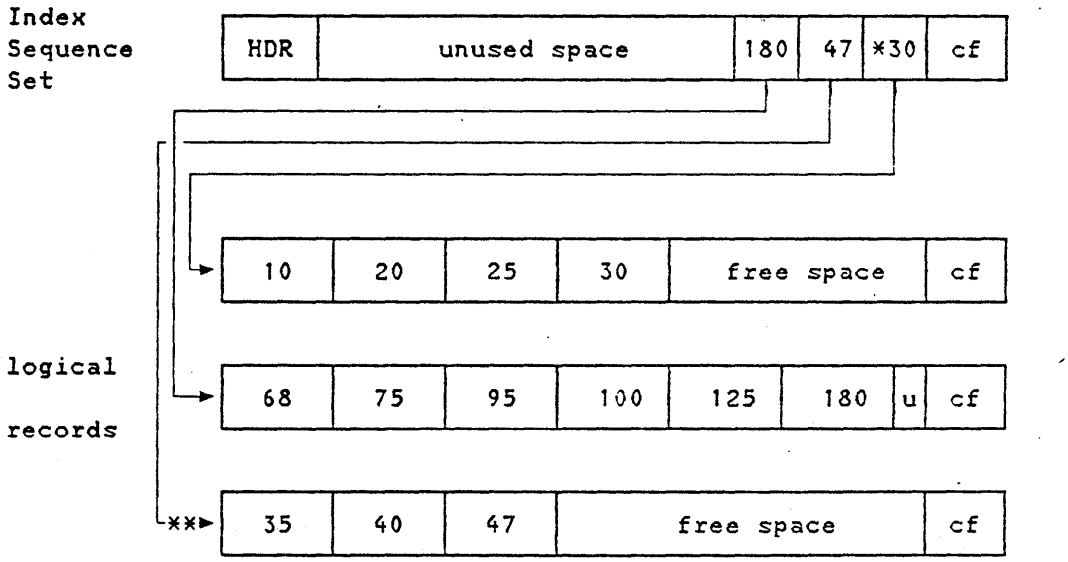


Figure 16. Control interval split (2 of 2)

3.2.1.1 VSAM Split-In-Progress Bit (Flag)

If an abnormal system termination occurred when processing a CI split, an integrity exposure existed. If the termination occurred after the updated sequence set record had been written and before the old data CI had been updated duplicate records existed in the data set. A PTF is available that fixes this problem (this problem is fixed in OS/VS MVS Rel. 3.8).

The solution consists of the following functions:

1. Indicate that a CI split is in progress by implementing a **'Split-In-Progress' (SIP) bit** (also called **CIDFBUSY flag**) in the CIDF of the data CI.

2. Detect an incomplete CI split by always checking the **SIP bit** for KSDS read operations.

3. Correct the data records involved in the incomplete split if necessary.

3.2.1.2 IMS/VS Use of VSAM Split-In-Progress Bit (Flag)

The integrity exposure mentioned above was always detected in an IMS/VS environment. This is because Data Base Backout Utility and Emergency Restart can determine if an INSERT operation did complete successfully. This is done by checking the presence of a corresponding log record type '52' followed by a log record type '50'. Log record '52' is created before **split start**, and log record '50' indicates **split end**.

If the log record '50' is missing, the message **DFS964 INSERT FAILED ON PRIOR UPDATE. RECOVERY REQUIRED FOR DATA BASE bbbbbbbb** is issued. A Data Base Recovery would be performed followed by a Data Base Backout operation.

With the implementation of the **SIP bit** there is no need for a Data Base Recovery operation and the message mentioned above is no longer issued. Data Base Backout Utility and Emergency Restart, when discovering that a log record '52' is not followed by a corresponding log record '50', will ask for the inserted segment by key. If not found, a new Buffer Handler call (READ BACKWARD) is issued. These requests will force VSAM to check the **SIP bit** in the CIDF.

If a 'Split-In-Progress' condition exists, VSAM will complete the CI split before returning control to the Data Base Backout Utility or Emergency Restart. Note that this only involves removing inconsistencies between data and index. VSAM cannot insert the record which initiated the split if the I/O to write it to DASD was not completed before the interrupt.

3.2.2 Control Area Splitting

If there is no free control interval in the control area, the record is not inserted until a **control area split** took place to provide free control intervals. VSAM obtains a new empty control area from the end of the data set.

When the control interval / control area split was initiated by **direct insert** the whole control area is split by reading about half of the control area into virtual storage and writing it back to auxiliary storage into the new control area.

When the control interval / control area split was initiated by **sequential insert** the control area is split at the point the control interval is to be inserted.

3.2.3 Insert Strategies Effect on Splitting

The control interval split as described in the previous section will be performed in two different ways depending upon the insert strategy chosen. There are two possible insert strategies that may be used: direct and sequential.

For IMS/VS data bases, the choice of insert strategy is made by using the INSERT parameter on the **OPTIONS** statement as explained in section "OPTIONS Statement" on page 80.

Direct insertion implies that a single record is to be inserted at some random location in the data set.

Sequential insertion is intended for inserting a large number of records, in key sequence, at some location in the data set. This form of insertion is also known as **mass insert**.

3.2.3.1 Direct Insertion with Split

When making inserts in direct mode and a split is necessary, the CI is split on a record boundary at the midpoint and approximately half of the records are moved to a new control interval. This is the case described above in "Control Interval Splitting (Direct Insert)" on page 29 and is the standard method of insertion and splitting.

Figure 17 shows the result of inserting three records at the same location in a control interval using the direct insertion technique.

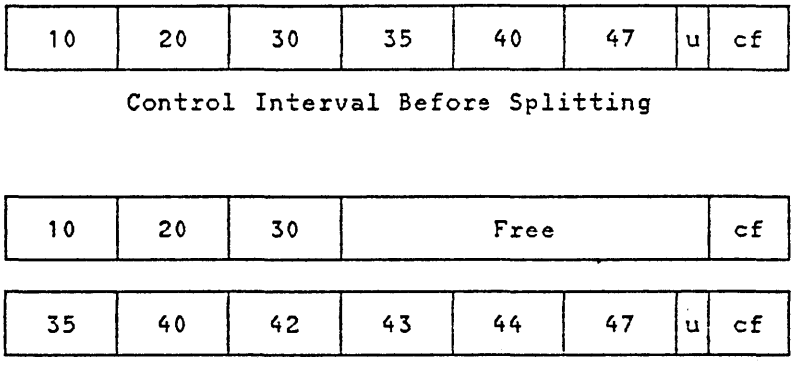


Figure 17. Direct Mode Insert Control Interval Split: Records 42, 43 and 44 are inserted into the control interval.

Direct insertion is highly suitable to random insert of records as, immediately following the CI split, each CI will have roughly 50% free space available for further insertions. This prevents the CI from having to be split again and again as new records are introduced into it. With genuinely random distribution of new records, this mode of processing is the most logical and serves its purpose very well.

However, in looking at Figure 17, it is apparent that if record 45 were now inserted another CI split would occur. Should a large number of records be inserted in sequence at this point in the data set, direct mode insertion will result in a CI split for every 4 records that are inserted. This is because 3 records from the old CI are carried forward to the new CI with each split due to splitting at the midpoint. Direct mode insert is therefore not very appropriate when mass inserts are being performed.

3.2.3.2 Sequential Insertion with Split

For mass inserts, the sequential insert technique is the best method. In sequential insert mode, the CI split occurs at the point of insertion and only those records following the one inserted are moved into the new control interval. The purpose of this is to allow a greater amount of free space at the insert position so that more records can be accepted before another CI split becomes necessary.

Figure 18 shows the effect of inserting a group of records into a particular position in the data set with sequential insert.

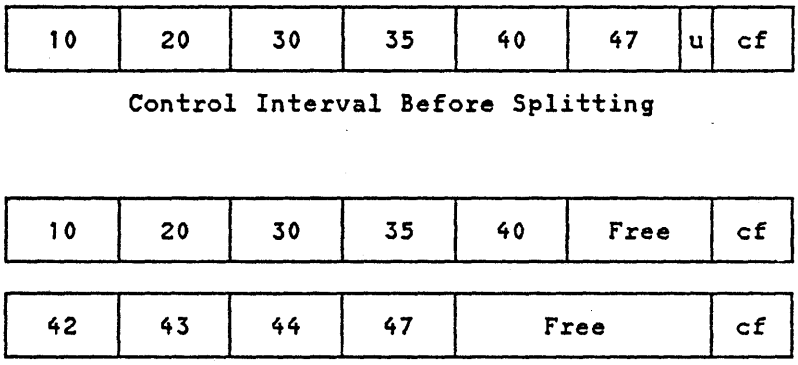


Figure 18. Sequential Mode Insert Control Interval Split: Records 42, 43 and 44 are inserted into the control interval.

The effect of sequential mode insert is very different from that of direct mode. For instance, in Figure 18 it can be seen that if record 45 were to be inserted no CI split would be necessary. Should insertion continue at this point, a CI split will occur after each 6th record inserted rather than after each 3rd as was the case for direct insert. Sequential mode is thus much more suitable to the insertion of blocks of records than is direct mode.

On the other hand, if random inserts are performed using sequential mode insertion, then the incidence of CI splits may increase. This is because the free space remaining in each CI after split is not equally distributed across the two CIs as it is with direct insert. One of the control intervals could be left with little or no free space following the split causing it to split again as soon as another insert into it is attempted. In extreme cases, this situation could lead to a CI split occurring for nearly every insertion.

3.3 CONTROL INTERVAL SIZES

3.3.1 Data CI Size

Several factors are important when choosing the best Data CI size:

- **Direct processing**: if direct processing is the predominant manner of accessing the data, then a choice of a small data CI is preferable. In general, select the smallest data CI that yields a reasonable space utilization. Normally 1024 or 2048 byte CIs are good, but on a device with large cylinder and track capacities like the IBM 3350 device, the index CI size will be 4 K when using 1 K data CIs.

- **Sequential processing**: if the processing is predominantly sequential, even larger data CIs may be a good choice.

3.3.1.1 DASD Utilization

A given logical record size may fit some CI sizes better than others. Generally, large CI sizes provide better fits. Also some CI sizes fit a track of a given device better than others.

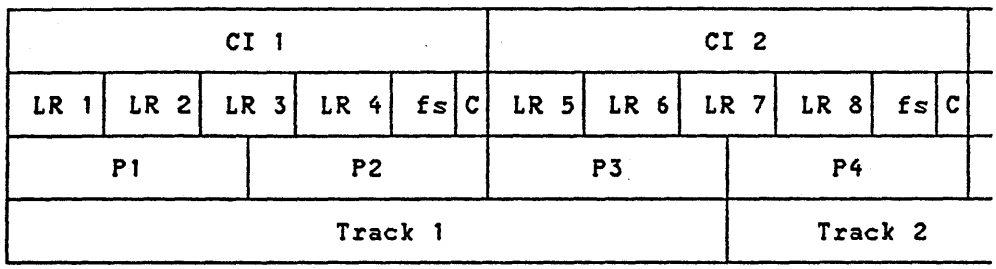
Each control interval is divided into one or more fixed length **physical records** and VSAM determines their size based on the control interval size and the device characteristics. This physical record is dependent on the control interval size and the device type and is only a consideration when mapping control intervals to the storage device. A physical record is that record seen on the disk and its size is equal to the control interval wherever possible.

The **physical record** sizes are:

512, 1024, 2048 and 4096 bytes.

Each control interval is automatically divided by VSAM into an integral number of physical blocks (e.g. CI = 512 results in physical block = 512, CI = 1536 results in physical block = 512, so 3 blocks are needed to hold one CI; CI = 6144 results in physical block = 2048, so 3 blocks are needed). By defining the CI size for a data set stored on a specific type of direct access device, the user implicitly chooses the physical record size.

Figure 19 on page 36 shows the relationship between user defined control interval size and VSAM chosen physical record size. In this example it is assumed the control interval size is 8K. As a control interval must be stored in an integral number of physical records the next lower value is 4K. So VSAM chooses 4K as physical record size for the 3330. Note that a logical record may span several physical records, but a physical record cannot span 2 tracks.



C = Control Fields (RDFs,CIDF) CI = Control Interval
fs = free space LR = Logical Record
P = Physical Record

Figure 19. Logical to physical records relationship

The following figure shows the number of physical records per track:

Device characteristics			Number of physical records/track			
Device	Trk/Cyl	Cyl/Vol	512	1024	2048	4096
3330-1	19	404	20	11	6	3
3330-11	19	808	20	11	6	3
3350	30	555	27	15	8	4

Figure 20. DASD track capabilities

For example, on an IBM 3350 track a 512 byte CI yields a potential 13824 bytes (27 CIs) (see Figure 20), which is 84% of VSAM-usable space per track whereas a 4096 byte CI yields 16384 bytes (4 CIs) of data on a IBM 3350 track, which is 100% of VSAM-usable space. Assuming a 80 byte record; in one case there would be 162 records per track and in the other case there would be 204 records per track.

3.3.2 Index CI Size

3.3.2.1 Relation of Index CI Size to Data CA Size

If no index CI value is defined for a KSDS data set (which is suggested) VSAM calculates the INDEX CI size based on the number of data CIs/CA and an estimated key compression value. The VSAM chosen index CI size may be checked using the LISTCAT command (see "INDEX Component Attributes" on page 130). The user may, however, define an index CI up to 4096. This large CI size may help to have fewer index levels, as the highest level index CI can hold a maximum of about 502 entries.

An index control interval may hold the following number of entries (8 bytes/entry based on normal key compression (see end of section "Key Compression" on page 43):

CI Size	Number of entries
512	58
1024	121
2048	248
4096	502

Figure 21. Average number of entries per INDEX CI

The size of an index CI also affects the replication factor (when REPLICATION or IMBED is used) and thus affects performance. Figure 20 on page 36 shows that a CI with a size of 512 bytes may be repeated 27 times on a 3350 track while a CI with a size of 4096 bytes may only be repeated 4 times. Therefore, the wait time to access a 512 byte CI is shorter than that of a CI with a size of 4096 bytes. For further discussion of REPLICATION and IMBED see "IMBED, REPLICATE" on page 76.

3.3.2.2 Problems of Incorrect Data CA Sizes / Index CI Sizes

If you specify an index CI size which is too small to hold all necessary entries for all CIs of one CA, the CA's are never filled to their maximum. In general it is better to let VSAM choose the index CI size. After DEFINE, a LISTCAT command can be issued and the entries be checked to redefine the cluster with a different data CA size if the index record is too big (see "Statistical Information" on page 125).

3.4 VSAM BUFFERS AND BUFFER POOLS

3.4.1 VSAM Buffer Use

Nonshared resources is the standard usage of VSAM buffers for one data set only. VSAM buffers are used by VSAM to read/write CIs from/to DASD. To increase performance, there are parameters (three for KSDS and two for ESDS and RRDS), to override the VSAM default values:

BUFNI number of index buffers (default 1) (KSDS only)
BUFND number of data buffers (default 2)
BUFSP amount of virtual storage to be reserved for VSAM buffers, when opening the data set (default = 2 data buffer and 1 index buffer)

BUFND and **BUFNI** are ACB parameters (ACB macros are used in Assembler programs) and, in OS/VS systems, are also JCL parameters. **BUFSP** is a DEFINE, ACB and JCL parameter. Any combination greater than the default of **BUFNI** and **BUFND** can be used.

If one of the parameters is specified in the ACB macro instruction (in the user program) this value overrides the default values, or in case of **BUFSP** the DEFINE value, **even if it is a smaller value**. If a parameter is specified in JCL, this value overrides any previous specification of this parameter (ACB or DEFINE). A detailed description of VSAM buffer usage is included in **OS/VS VSAM SHARING, G320-6015.**

3.4.1.1 Data Buffers

To calculate the number of data buffers, the number of **strings** (STRNO) used must be determined. A string is a request to a VSAM data set requiring data set positioning. VSAM stores (e.g. for a sequential access) the information about which record has been accessed. If different concurrent accesses to the same data set are necessary **multiple strings** are used. Multiple strings allow, for example, access to the data set directly with one string and sequentially with the second string. For each string there must be at least one buffer.

Usually the user defines the amount of strings he intends to use in his program (ACB STRNO=x). But when all strings are active and there is another request waiting, a new string is built dynamically. **Dynamic string addition** does not apply to shared resources as used in CICS and IMS. The string is not erased, when it is not used any more, but it can be reused for the next request.

The minimum number of data buffers is two (1 per string + 1 for splits). It is very important to know that the OS/VS user can use only one of these two data buffers. One data buffer is always reserved by VSAM for splits. If 4 data buffers should be used, 5 must be specified.

3.4.1.2 Index Buffers

The minimum number of index buffers is one. It may be advisable to have more index buffers for extensive index operations (direct). If multiple strings are used, the minimum needed is one index buffer per string.

Generally, the more index buffers, the better performance that can be obtained. Data buffers are not important for direct access, because only one is used for each access. As there is always a top down search through the index, at least two index buffers should be specified, to hold the highest index CI in the buffer. If more than one index buffer is specified, the additional buffers can hold index set CIs but only one index buffer is used for the index sequence set per string.

If too few index buffers are specified, the result may be poor performance, as extensive EXCPs may have to be issued. If there are three index levels and only one index buffer, three index EXCPs are necessary to find the pointer to the correct data CI.

Suggested number of buffers:

BUFNI: minimum = no. of index levels * STRNO (default = 1)
maximum = (no. of records (CI's) in index set + 1) * STRNO
BUFND: default = 2 (STRNO value + 1 for splits) (STRNO default is 1)

General suggestion: Many data buffer for sequential processing and many index buffers for direct processing.

3.4.2 Buffer Pools

The DL/I buffering services are controlled by three pools of control block and buffers:

1. the OSAM buffer pool
2. the DL/I buffer handler pool
3. the VSAM buffer pool

The DL/I buffering services are the interface between the DL/I action modules (for example, Retrieve, Delete, Insert) and the data management access methods (VSAM and OSAM). Whenever an action module needs to inspect or change data in a data base, buffering services are called to perform whatever physical reading or writing is required. A separate pool of buffers is allocated for each type of data base: VSAM and OSAM. Data bases that use the VSAM access method share the use of buffers in the **VSAM shared resource pool**. Data bases that use the OSAM access method share the use of buffers in the **OSAM buffer pool**. The **DL/I buffer handler pool** is the focal point for recording buffering services activity. The DL/I buffer handler pool and the OSAM buffer pool are further described in **IMS/VS VERSION 1 PRIMER, SH20-9145** in chapter 7 'Installing IMS/VS '.

The requirement for a resource pool is generally associated with a program having access to a large number of data sets OPENed concurrently. The environment of such a program creates a heavy, but varying, demand of data from external storage. DB/DC systems such as IMS/VS and CICS/VS, are typical environments for use of a shared resource pool.

The concept of the buffer pool allows blocks of data to remain in main storage as long as possible, in order to avoid secondary storage reads and writes. Data in the buffer pool can be accessed and updated without causing I/O operations as long as there is no need to reuse the buffer space the data occupies. A use chain determines the order in which the buffers are used. Empty buffers are placed at the bottom of the use chain and are always available for reuse.

As buffers are accessed they are placed at the top of the use chain. When a retrieve request occurs, the buffer pool is searched using the use chain, to determine if the requested data is already in main storage. If the data is not found, the least recently used buffer (bottom of the use chain) is selected, the old data is written out if it has been changed and the requested data is read into the selected buffer. The buffers are pooled according to CI size instead of usage (e.g. data or index).

3.4.2.1 VSAM Shared Resources

This feature provides the user with a way of sharing the VSAM I/O related control blocks, buffers, and channel programs storage among many VSAM data sets. This is contrasted to the way in which these resources are managed by data set in **Nonshared Resources (NSR)**. The IMS/VS community often refers to NSR as 'native VSAM'. With NSR, storage for I/O operations is allocated to a particular data set. Therefore, with NSR, an I/O buffer can only be used for transfer of data to and from a single data set.

With NSR, a Sequence Set record cannot be shared among multiple concurrent requests, each has his own copy. Also, the data buffer would only be reused if the subsequent request was assigned the same PLH (Placeholder) as the previous request to the same data control interval, and this only happens if the buffer is still valid. In multi-string processing with NSR, it is highly improbable that a data buffer will be reused and additional physical I/O would be scheduled. This I/O could be eliminated by using Shared Resources.

In Shared Resources (Local Shared Resources (LSR) and Global Shared Resources (GSR)), this same buffer can contain data from data set 'A' at one time and subsequently this same buffer can be used for transfer of data to and from data set 'B'. The same technique applies to storage allocated for I/O related control blocks.

The storage for resource sharing can be acquired for use by a single partition or address space. This is referred to as **Local Shared Resources (LSR)** and can be specified for all OS/VS Operating Systems. The LSR buffer pool can be shared by all VSAM data sets within that partition/address space(region).

In MVS only, common storage can be acquired which allows all address spaces to share one resource pool. This is referred as **Global Shared Resources (GSR)**. In an MVS system, a single address space can have both a global and a local resource pool and tasks may use either pool for resources. However, a data set must be specified to get its resources from only one of the pools.

3.4.2.2 IMS/VS usage of VSAM Shared Resources

With the exception of Fast Path data bases, IMS/VS uses VSAM Shared Resources for online data base processing, and provides complete integrity support for all data base users within that control block and buffer structure. For Fast Path, IMS/VS uses a VSAM facility called 'Control Blocks in Common' (see **OS/VS VSAM OPTIONS FOR ADVANCED APPLICATIONS, GC26-3819**), which provides a system-wide set of control blocks even though the VSAM processing is Nonshared Resources.

Message and batch message regions will be able to 'share' the same data base with full integrity, but equivalent 'sharing' between batch regions or between batch and online is not supported, regardless of whether the

batch users are IMS/VS or nonIMS/VS. The official method for bringing batch work into the IMS/VS integrity scheme is by running batch applications as batch message regions.

DL/I maintains statistics on buffer pool utilization and access method requests. These statistics can be used to determine the optimum buffer pool sizes for a job. The buffer pool statistics are described in "Data Base Statistics" on page 133.

The following is a summary chart to show which type of VSAM Shared/Nonshared Resources is used by IMS/VS.

<table>
<tr><td rowspan="2">ONLINE</td><td colspan="3" rowspan="2">MVS</td><td>GSR (one only)</td></tr>
<tr><td>LSR</td></tr>
<tr><td></td><td colspan="3">VS1</td><td>LSR</td></tr>
<tr><td rowspan="8">BATCH</td><td colspan="3">USER</td><td>LSR</td></tr>
<tr><td colspan="3">DB RECOVERY:
- IMAGE COPY alone
- IMAGE COPY with CHANGE ACCUM. TAPE
- CHANGE ACCUMULATION TAPE alone
- LOG TAPE</td><td>
NSR
NSR
LSR
LSR</td></tr>
<tr><td colspan="3">IMAGE COPY</td><td>NSR</td></tr>
<tr><td colspan="3">Other UTILITIES</td><td>LSR</td></tr>
<tr><td rowspan="4">USER
INITIAL
LOAD
PROGRAM</td><td colspan="2">SHISAM</td><td>NSR</td></tr>
<tr><td rowspan="2">HISAM</td><td>ESDS</td><td>LSR</td></tr>
<tr><td>KSDS</td><td>NSR</td></tr>
<tr><td colspan="2">Others[1]</td><td>LSR</td></tr>
</table>

GSR/LSR/NSR = Global Shared / Local Shared / Nonshared Resources

Figure 22. IMS/VS usage of VSAM Shared/Nonshared Resources

The data set related buffer options (BUFSP, BUFND, BUFNI, see "VSAM Buffer Use" on page 37) are only in effect when IMS/VS uses the NSR buffer management technique. These specifications are ignored by VSAM OPEN when a data set is directed to the pool (LSR/GSR) for these resources.

---

1 including INDEXES (primary and secondary)

3.4.2.3 Relation of CI Sizes to IMS/VS Buffer Sizes

The following Shared Resource Pool buffer sizes can be specified in bytes:

512 1024 2048 4096 8092 12288 16184 20480 24576 28672 32368

A maximum of 255 buffers per size may be specified.

The minimum number of buffers is 3 for HDAM load and HDAM/HIDAM processing (one PCB), or 4 for HISAM/HIDAM load and HISAM processing (one PCB).

The Resource Pool buffer size specifications are directly related to the VSAM CI size used in the data sets sharing the pool. When using 4096 bytes CI size, buffers with a size of 4096 (or larger) are to be specified.

For any CI size different from the possible buffer pools sizes the next larger size must be specified; i.e. if a CI size of 6144 bytes is used, buffers with 8192 bytes must be specified, even when the 2048 excess bytes in the buffer are not used. Therefore, it is advisable when using Shared Resources to specify CI sizes which correspond to the allowed buffer pool sizes.

3.4.2.4 Buffer Allocation Considerations

There is no generalized number of how many buffers should be specified, since there are many aspects to consider, e.g. how many concurrent requests will be active against the entire pool at the peak period, or what are the limits of the main storage to be used for the pool.

One general rule, however, should be considered:

Keep control interval sizes consistent within index components and control interval sizes consistent within data components, but different from index components.

When following this rule, index control interval buffers are not reused when extra data buffers are needed. The probability is much higher that an index control interval can satisfy multiple requests than that of a data control interval. It would, therefore, be a better use of system resources to allocate more real storage for index buffers than for data. Again, this can only be accomplished by controlling the control interval size(s) of the data set component(s).

Further detailed information is included in **OS/VS VSAM SHARING, G320-6015.**

3.5 KEY COMPRESSION

As described in section "KSDS" on page 9 the index set contains an entry for each index record on the next lower level and the index sequence set contains an entry for each data CI of a KSDS. Each entry consists of the highest key in that CI and an RBA offset. To shorten the size of the index entry, the keys are compressed by VSAM routines. This is called key compression. The key compression uses front key compression and rear key compression. The following examples show key compression in the sequence set.

3.5.1 How Compression Works

Assuming the following CIs exist in a cluster:

CI 1	10001	10002	10003	10009
CI 2	10052	10060	10070	10080
CI 3	10222	10250	10300	10333
CI 4	14021	14023	14024	14028

Figure 23. Key compression (1 of 5)

For front key compression the highest key of the CI is compared to the highest key preceding it in the index, and all identical values starting from the left are compressed:

Full key	Front compressed key	
00000		(lowest possible key)
10009	10009	
10009		
10080	===80	
10080		
10333	==333	
10333		
14028	=4028	

Figure 24. Key compression (2 of 5)

Exceptions to the **front key compression** rule:

1. The key in the first entry in an index record is not front compressed.
2. The highest key in an index record section (a subdivision of an index record) is compared with the highest key in the preceding section to allow a 'binary search' in the index record.
3. The index entry to the right of the first section pointer is not front compressed.

For **rear key compression** the highest key of the CI is compared with the lowest key of the **next** CI, and so on. The characters to the right of the first unequal character are eliminated:

Full key	Rear compressed key
10009	1000-
10052	
10080	100--
10222	
10333	10---
14021	
14028	14028

Figure 25. Key compression (3 of 5)

VSAM uses **both** types of compression. To indicate how many bytes are compressed and how large the compressed key is, two one byte fields are used. The 'F' field contains the number of front key compressed bytes. The 'L' field contains the residual key length.

Combining front and rear key compression the compressed keys and their F+L fields are as follows ('=' is front and '-' is rear key compression):

CI No.	Full key	Full compressed key	F	L
CI 1	10009	1000-	0	4
CI 2	10080	===--	3	0
CI 3	10333	==---	2	0
CI 4	14028	=4028	1	4

Figure 26. Key compression (4 of 5)

VSAM reconstructs a compressed key as follows:

1. Front compressed values are taken from the previous uncompressed key.

2. Rear compressed values are substituted by X'FF'.

The previously compressed keys would then be reconstructed as follows ('f' represents the hex value X'FF'):

CI No.	Full key	Full compressed key	F	L	reconstructed key
CI 1	10009	1000-	0	4	1000f
CI 2	10080	====-	3	0	100ff
CI 3	10333	==---	2	0	10fff
CI 4	14028	=4028	1	4	14028

Figure 27. Key compression (5 of 5)

Due to rear key compression the reconstructed key shows the potential highest key that could be placed in the CI and not the actual highest key currently stored in the CI.

Based on the key compression, VSAM determines, into which CI a new record has to be inserted. A record with the value 10088 would be inserted into CI 2 since the value is lower than 100ff (f=X'FF'). A record with the value of 10100 would be inserted into CI 3.

Single field keys do compress well. There is a great chance that larger keys (20-30 bytes) compress out to 8 or 9 bytes including control information. Smaller keys (5-15 bytes) may compress to 3 to 5 bytes.

Assuming a key would compress to 3 bytes then an index entry would have a length of 6 to 8 bytes which consists of the following:

- 1-3 bytes pointer to CI. The length of the pointer depends on the number of CIs to be referenced in that index record:

  1 byte = less than 256
  2 bytes = 256 to 65535
  3 bytes = 65536 or more

- 1 byte 'F' (number of front key compressed bytes)

- 1 byte 'L' (residual key length)

- 3 bytes compressed key

3.5.1.1 Problems with Poor Compression

When keys do not compress well, each index entry is larger with the result that fewer entries can be placed in each index control interval. This means that the sequence set records cannot point to as many data CIs and that each index set record will govern fewer lower level index records.

It may be found that when loading records into the data set, the sequence set record will become full before all of the CIs in the corresponding control area have been loaded. When this occurs, VSAM cannot continue to load records into this CA and has to begin loading a new CA with a new sequence set record. The data set will then contain a number of CIs in each CA which cannot be used in any way. This means that a certain percentage of the DASD space occupied by the data set is effectively wasted and the data set may go into a secondary extent sooner than is expected.

To determine whether one is experiencing bad compression, one CA's worth of records should be loaded and then the data set's catalog information listed. If it is found that more than one CA and one index record have been used then bad compression is wasting space in the data set.

Another difficulty with poor compression is that since the high level index records can only control a small number of lower level records, the index may have to contain more levels than it would ordinarily. This will have an adverse impact on random processing of the data set as there will be more index levels through which the search must pass.

These problems may be addressed in several ways:

- increase the size of the index control intervals in order to allow for the bad compression
- increase the data control interval size to reduce the number of CIs per CA (therefore less index entries)
- redesigne the keys so that they will compress properly

Increasing the size of the index control intervals will cause a greater demand for buffer space and make it more difficult to allocate sufficient buffers to achieve good performance. On the other hand, it may not be possible to alter the key structure without making large scale changes in the applications. If possible, it is probably better to change the key rather than to increase the index CI size because of the impact of CI size on performance.

3.5.1.2 Characteristics of Keys That Do Not Compress Well

Since bad key compression can cause severe difficulties, users of VSAM should be aware of the characteristics of keys that compress badly so that these problems can be avoided at the design stage. In general, keys do compress nicely if the high order portion of the key is very stable so that many keys have the same leading characters and the low order portion of the key changes frequently so that there are large differences in the rightmost characters.

Poor key compression will occur if the keys have some combination of the following characteristics:

- Keys are made up of multiple contiguous fields.
- Changes occur in the high and low parts of the key but not in the middle. The worst case is for a multiple key field with changes in the high order portion of the first field and the low order portion of the last field.
- The keys fall into groups where the number of keys in a group is less than the number of records in a data CI. This means that the high keys in each data CI will not have the same leading characters and front compression will be almost nonexistent.
- The last field of the key is long and very dense so that it does not change to a very great extent. Since rear compression is best when there are large differences in the low order part of the keys, this will lead to very little rear compression being done.

As an example of a key that compresses badly consider:

NNNXXXXXXXXXXSS

NNN is a field which changes every 4 or 5 records and there are more than 5 records in each CI.

XXXXXXXXXX is a field which does not change or changes very rarely.

SS is a field which changes for every record but cycles within the changes of the NNN field.

If NNN were to change every 20 - 25 records then compression would improve. It would also be better if SS seldom changed or continued to change but was placed next to NNN so that the key structure would be: NNNSSXXXXXXXXXX.

Note that VSAM determines the index CI size at DEFINE time while key compression takes place at LOAD time. Therefore, VSAM is not able to calculate the optimum index CI size.

3.5.1.3 Criteria for Key Selection

There are three basic criteria to be observed when selecting keys:

1. Keys must be unique over the entire data set.

2. Avoid keys which will not compress well.

3. Avoid keys which may lead to crawling (see description in section "Crawling" on page 102).

The best key is a single field which is unique in each record and whose values are randomly distributed. Often, however, this kind of key cannot be found or is not feasible as a search argument for the applications.

Keys may have to be constructed from multiple contiguous fields in order to achieve uniqueness or in order to meet the needs of the applications. In this case the user should try to set up the key so that its high order portions are the most stable and its low order portions are volatile so that it will compress properly.

If a date is to be used as part of the key, it is best to store the date in the form, YY/MM/DD, to assist compression.

4.0 RELATIONSHIP OF IMS/VS AND VSAM PARAMETERS

There are many parameters in the definition of IMS/VS data bases and VSAM clusters which either require corresponding specification or affect the performance of the system.

This chapter is intended to point out those cases where there exists a correspondence between DBD generation and Cluster definition and also to highlight those parameters which are performance oriented.

4.1 DATA BASE DEFINITION PARAMETERS

This section is concerned with the VSAM-related parameters of the DBD specification. Not all of the parameters of the Data Base Definition are covered because the discussion is restricted to those parameters which either require VSAM or have an effect on the definition of the VSAM clusters which will comprise the data base.

The parameters which are treated are not always dealt with in full detail since the objective is to cover only the VSAM-related aspects of the DBD generation.

For a complete description of all of the parameters used in DBD generation the reader should refer to the IMS/VS UTILITIES REFERENCE MANUAL, SH20-9029.

4.1.1 DBD Macro

The following parameters are part of the specification of the DBD macro which is the first macro used in the DBD generation.

```
ACCESS=(DL/I access method, Operating System access method)
RMNAME=(module name,anchor points,maximum block number,bytes).
PASSWD=YES or PASSWD=NO.
```

Figure 28. DBD macro parameter summary

4.1.1.1 ACCESS

The format of this parameter is:

ACCESS=(DL/I access method, Operating System access method)

The '**DL/I access methods**' which may be specified are: HSAM, SHSAM, HISAM, SHISAM, HDAM, HIDAM, GSAM, INDEX, LOGICAL, MSDB and DEDB.

No'**Operating System access method**' may be specified when the DL/I access method is either HSAM, SHSAM, LOGICAL, MSDB or DEDB. With the release of IMS/VS 1.1.1, VSAM was made the default '**Operating System access method**' for the following DL/I access methods: HISAM, HDAM, HIDAM and GSAM. Thus, unless some other access method is explicitly requested, VSAM will be assumed for these four DL/I organizations.

For SHISAM and Secondary Index INDEX data bases it is required that VSAM be the '**Operating System access method**'. Any other specification with SHISAM or secondary index INDEX data bases is invalid.

DB-TYPE	DEFAULT	ALTERNATE	COMMENTS
HSAM	none	N/A	No choice allowed
SHSAM	none	N/A	No choice allowed
HISAM	VSAM	ISAM	
SHISAM	VSAM	invalid	VSAM is required
HDAM	VSAM	OSAM	
HIDAM	VSAM	OSAM	
LOGICAL	none	N/A	Not used for Logical
GSAM	VSAM	BSAM	
MSDB	none	N/A	Not used for MSDB
DEDB	none	N/A	Not used for DEDB
INDEX Primary	VSAM	ISAM	Applies to primary index of HIDAM DB
INDEX Secondary	VSAM	invalid	VSAM required for secondary indices

Figure 29. DL/I Organizations and Operating System Access Methods: The column headed **ALTERNATE** indicates the alternative Operating System access method that may be used if the default is not desired.

4.1.1.2 RMNAME

The RMNAME parameter is only valid when specified for HDAM or DEDB data bases. Therefore, the only circumstances when this parameter will have any effect on VSAM is when it applies to a HDAM data base using VSAM as the Operating System access method. The parameter is specified as:

RMNAME=(module name,anchor points,maximum block number,bytes).

The 'module name' subparameter specifies the name of the user-supplied randomizing module used to store and access root segments in the data base.

The 'anchor points' subparameter determines the number of root anchor points to be allowed in each control interval of the root addressable area of the HDAM data base. This specification affects the definition of the VSAM cluster since each anchor point will occupy four bytes in the control interval. This fact must be taken into account when determining the size of the control intervals to be used for the data base.

The 'maximum block number' subparameter specifies the size of the root addressable area for the HDAM data base. For VSAM, this value specifies the number of control intervals to be allocated to the root addressable area. It must be kept in mind that this is not the total number of control intervals in the data base. It applies only to the root addressable portion of the data base and it is still necessary to determine the space required for the overflow portion of the data base.

The 'bytes' subparameter has no special effect in regard to VSAM. It specifies the maximum number of bytes of a data base record that can be placed into the root addressable area with an unbroken series of insert calls.

4.1.1.3 PASSWD=YES

This parameter is only available for use with VSAM data bases and is invalid if used for data bases whose Operating System access method is ISAM, OSAM or BSAM.

The format of the PASSWD parameter is:

PASSWD=YES or PASSWD=NO.

The default value for this parameter is NO.

If PASSWD=YES is specified, then the Master- or Control-level password of the corresponding VSAM cluster(s) must be the same as the name of the DBD. The DBD name is specified as 1 to 8 alphameric characters with the NAME parameter of the DBD macro.

An explanation of the meaning, usage and definition of the various levels of VSAM passwords is given later in "Password Specification" on page 66.

In the IMS/VS DB/DC (online) environment, all OPENs for VSAM clusters will bypass password checking. This avoids the necessity of prompting the operator to supply the password.

For a DB-only (batch) system, VSAM password checking will be done. The operator will only be prompted to supply the password if PASSWD=NO has been specified and the VSAM cluster(s) are password-protected with passwords which are **not** the same as the DBD name.

If PASSWD=YES has been specified and the VSAM passwords are not the same as the DBD name, then a VSAM OPEN error will occur and the data base will not be opened.

The intended purpose of this parameter is to prevent accidental access of the IMS/VS data bases by nonIMS/VS programs. It is recommended that this facility be used to supply this protection.

4.1.2 DATASET Macro

The DATASET macro is used to describe the data sets that comprise the data base. This macro has several parameters which relate directly to the definition of the VSAM clusters and it is therefore very important that the user be aware of these relationships.

```
DD1=DD-name
OVFLW=DD-name
BLOCK=(blocking factor 1, blocking factor 2)
BLOCK=size
RECORD=(record length 1, record length 2)
SIZE=(size 1, size 2)
FRSPC=(Free Block Frequency Factor,Free Space Percentage Factor)
RECFM=F|FB|V|VB
UOW=(number 1, overflow 1)
ROOT=(number 1, overflow 1)
```

Figure 30. DATASET macro parameter summary

4.1.2.1 DD1

The **DD1** parameter specifies the 1- to 8-character DD-name which will be used in the JCL to reference the data set described by this macro. The format of the **DD1** parameter is as follows:

```
DD1=DD-name
```

For VSAM data sets the format of the DD statement should be:

```
//DD-name  DD DSN=clustername,DISP=SHR.
```

'DD-name' must be the same name as specified for the value of the **DD1** parameter of the DATASET macro.

'clustername' is the name of the VSAM cluster as specified in the cluster definition. The **DD1** parameter and the JCL DD-statement form the linkage between the data base and the VSAM cluster. This is the only means of relating the data base to the VSAM cluster.

4.1.2.2 OVFLW

The only cases in which the **OVFLW** parameter should be used with VSAM data bases is when the data base is either HISAM with more than 1 segment type or it is an INDEX data base for a secondary index using nonunique keys. Other than these two cases the use of **OVFLW** is invalid for VSAM data bases.

The format of the **OVFLW** parameter is as follows:

```
OVFLW=DD-name
```

This parameter is similar to the **DD1** parameter. The value of **'DD-name'** is also a 1- to 8-character name and the format of the referenced DD-statement is the same as that shown above for **DD1**. It is used to reference the VSAM ESDS cluster used as the overflow component of the data base.

4.1.2.3 BLOCK, RECORD and SIZE

These three parameters of the DATASET macro are used to specify the size of DASD storage units to be used for the data base. In the case of VSAM data bases, these DASD storage units are Control Intervals and the specification must correspond to the Control Interval size defined for the VSAM cluster. There are no specific default values for these parameters. If no values are given for these parameters, then DBD generation will calculate the values to be used.

The **BLOCK** parameter is used either to specify the blocking factors or to specify the block size directly. The **RECORD** parameter defines the record sizes in the data base. The **SIZE** parameter is used to override the computation that is performed by DBD generation by directly indicating the control interval size that is to be used.

There are two formats which may be used for the **BLOCK** parameter:

BLOCK=(blocking factor 1, blocking factor 2)
BLOCK=size

The **'blocking factors'** indicate the number of records to be included in each unit of DASD storage. Depending upon the data base organization used, **'blocking factor 1'** and **'blocking factor 2'** will have different areas of applicability as shown in Figure 31. If the second form of the parameter is used, it indicates the size of the data portion of the block without DL/I or VSAM overheads.

ACCESS METHOD	Blocking Factor 1	Blocking Factor 2
GSAM	Input/Output[2]	invalid
HISAM	Primary (KSDS)	Overflow (ESDS)
SHISAM	Primary (KSDS)	N/A[3]
HDAM	Use 'Block=size' format[4]	
HIDAM	Use 'BLOCK=size' format[4]	
INDEX	Primary (KSDS)	Overflow (ESDS)
DEDB	invalid	

Figure 31. Relation of BLOCK subparameters to Access Methods: The table shows the data sets to which the **'blocking factor 1'** and **'blocking factor 2'** values apply for the various data base access methods.

---

2 Applies to both the input and output data sets of the GSAM data base.

3 A SHISAM data base has no Overflow component and the **'blocking factor 2'** subparameter is therefore not used in this case.

4 The **'blocking factor 1'** and **'blocking factor 2'** subparameters are not applicable to the HD access methods so the second format of the **BLOCK** parameter must be used.

The format of the **RECORD** parameter is :

RECORD=(record length 1, record length 2)

As with the **BLOCK** parameter, **'record length 1'** and **'record length 2'** will have different areas of application depending on the data base organization that is used.

The relationship of the parameters to the data base organizations is shown in the following table.

ACCESS METHOD	Record Length 1	Record Length 2
GSAM	Fixed : LRECL Variable: Maximum	Fixed : N/A[5] Variable: Minimum[5]
HISAM	Primary(KSDS)	Overflow(ESDS)
SHISAM	Primary(KSDS)	N/A[6]
HDAM	ignored[7]	
HIDAM	ignored[7]	
INDEX	Primary(KSDS)	Overflow(ESDS)
DEDB	invalid	

Figure 32. Relation of RECORD subparameters to Access Methods: This table shows the data sets to which the **'record length 1'** and **'record length 2'** values apply for the various data base access methods.

---

5 This subparameter is only used when the GSAM data base contains variable length records. The value of **'record length 2'** must be less than or equal to the value of **'record length 1'** in this case.

6 A SHISAM data base has no Overflow component and it is therefore not used in this case.

7 The **RECORD** parameter has no effect when specified for either of the HD access methods.

The **SIZE** parameter has the following format:

SIZE=(size 1, size 2)

The area of applicability of **'size 1'** and **'size 2'** also varies depending on the data base organization as shown in the following table.

ACCESS METHOD	Size 1	Size 2
GSAM	Input/Output	invalid
HISAM	Primary(KSDS)	Overflow(ESDS)
SHISAM	Primary(KSDS)	N/A[8]
HDAM	Data(ESDS)	invalid[9]
HIDAM	Data(ESDS)	invalid[10]
INDEX	Primary(KSDS)	Overflow(ESDS)
DEDB	Data(ESDS)[10]	invalid

Figure 33. Relation of SIZE subparameters to Access Methods

Although there is a choice of whether to use the **BLOCK, RECORD** or **SIZE** parameters for the various data base organizations, there is a recommended parameter combination for each of the available organizations.

For a GSAM data base, the **BLOCK** and **RECORD** parameters should be used in conjunction. The control interval size that will be used for the GSAM data base is the product of the specifications for the blocking factor and the record length plus the system overhead rounded up to the next larger valid control interval size. The valid control interval sizes are described in section "Control Interval and Control Area Definition and Concept" on page 4. The system overhead included in the calculation of the control interval size is shown in Figure 34 on page 57.

For HISAM, SHISAM and INDEX data bases, it is suggested that the **SIZE** parameter be used instead of **BLOCK** and **RECORD**. DEDB data bases must be defined with the **SIZE** parameter. When **SIZE** is used, the value specified is the actual control interval size that is desired. This value must be equal to one of the valid control interval sizes. If it is not, then the value is rounded upwards to the next valid size and DBD generation will issue a warning message. Although there is no overhead added to the value speci-

---

[8] A SHISAM data base has no Overflow component and it is therefore invalid to specify the **'size 2'** subparameter for SHISAM'

[9] The **SIZE** parameter applies only to the data component of the HD data base organizations and the **'size 2'** subparameter is meaningless when specified for either of the HD organizations.

[10] The **SIZE** parameter is required for a DEDB data base.

fied for the **SIZE** parameter, the user must be aware of the overhead requirements in order to be able to select the appropriate control interval size. The overhead requirements for these data base organizations is shown in Figure 34.

When either HDAM or HIDAM data base are being defined the second form of the **BLOCK** parameter is to be used. The size specified is for the data (ESDS) component of the data base. In this case, the value specified for the **BLOCK** parameter is the amount of space required for the segments that are to be placed in each block. DBD generation will add to this value the space needed for root anchor points, a free space anchor point and the VSAM control information. The result is then rounded up to the next valid control interval size.

The amount of space required for the anchor points and VSAM control information is shown in Figure 34.

Access Method	Characteristics	VSAM Overhead	DL/I Overhead	Total Ovrhd
GSAM	Block = 1	3(RDF) + 4(CIDF)	0	7
GSAM / SHISAM	fixed rcd. Block > 1	2*3(RDF) + 4(CIDF)	0	10
GSAM	variable rcd. Block > 1	3*block.factor(RDF) + 4(CIDF)	0	>=10
HISAM		3(RDF) + 4(CIDF)	5	12
INDEX	unique key[11]	3(RDF) + 4(CIDF)	0	7
INDEX	nonunique key	3(RDF) + 4(CIDF)	4	11
HDAM		3(RDF) + 4(CIDF)	4 for FSAP 4 per RAP[12]	>=15
HIDAM		3(RDF) + 4(CIDF)	4 for FSAP[13] 4 for 1 RAP	11/15

Figure 34. VSAM and DL/I overhead to be used for CI size calculation

---

11 This case also applies to the Primary Index of an HIDAM data base which, by definition must have unique keys.

12 The number of Root Anchor Points (RAPs) in each control interval is determined by the value specified as the second subparameter of the RMNAME macro as explained in "RMNAME" on page 51.

13 A Free Space Anchor Point (FSAP) will only be used in an HIDAM data base if no pointers other than Hierarchical Forward and Physical Twin Pointers are specified for the root segment.

4.1.2.4 RECFM for GSAM/VSAM

This parameter applies only to GSAM data bases and may not be specified for any other data base organization.

The values of this parameter which may be used with a VSAM-based GSAM data base are:

F indicates that the records are fixed length.

FB indicates that the records are fixed length and blocked.

V indicates that the records are of variable length.

VB indicates that the records are of variable length and are blocked.

When the GSAM data base is specified as having variable length records, it is assumed that the first two bytes of the record contain the record length. This record length must be supplied by the user when creating new records and is supplied by DL/I when the records are retrieved.

For the fixed length record formats, the record length need not be specified. The use of blocked formats is recommended for VSAM unless the size of the GSAM data base records is such that only one record will fit into a control interval.

4.1.2.5 FRSPC

The **FRSPC** parameter is only applicable to HDAM and HIDAM data bases. When this parameter is used for a VSAM data base, it specifies the amount of free space to be allowed in the ESDS component of the data base at initial load time. The ESDS component of an HIDAM data base is that portion containing the actual data segments. For an HDAM data base, the parameter applies only to that portion of the ESDS which has been defined to be the root addressable area using the **RMNAME** parameter as explained in "RMNAME" on page 51.

This parameter has the following format:

FRSPC=(Free Block Frequency Factor,Free Space Percentage Factor)

The meaning and use of the the subparameters has been discussed earlier in "DL/I - ESDS Free Space" on page 24.

4.1.2.6 Fast Path Parameters

For FAST PATH data sets which **must** be VSAM data sets the parameters UOW (Unit-of-Work) and ROOT must be specified. A detailed description of their function and specification is included in chapter "Fast Path Feature" on page 103. The format of the UOW parameter is as follows:

```
UOW=(number 1, overflow 1)
```

The format of the ROOT parameter is as follows:

```
ROOT=(number 1, overflow 1)
```

4.1.3 Other DBD Parameters That Affect VSAM

Because the other macros in DBD generation do not affect VSAM in as direct a manner as do the **DBD** and **DATASET macros**, the remaining parameters for which there are VSAM considerations are treated as a group rather than macro by macro.

4.1.3.1 Pointer Specifications

The use of pointers in the data base organizations has a bearing on VSAM in that it will affect the size of segment prefixes. These segment prefixes must be included in the total size of the segments when making the determination of control interval sizes.

SEGM Pointer Parameters

There are two parameters of the **SEGM** macro that affect pointer use for HDAM and HIDAM data bases. The first of these is the **PARENT** parameter.

The format of this parameter is:

```
PARENT=((physprnt,pointer specification),(logprnt,P/V,logprtdb)
```

'physprnt' is the name of the segment which is the physical parent of the segment being defined by this **SEGM** macro. The pointer specifications that may be used with the physical parent are: SNGL or DBLE. If SNGL is specified one 4-byte pointer is included **in the prefix of the physical parent,** not in the prefix of the segment being defined. If DBLE is specified then two 4-byte pointers are included in the physical parent's prefix.

'logprnt' is the name of the segment which is the logical parent of the segment being defined and 'logprtdb' is the name of the data base in which the logical parent segment is defined.

The use of 'P' with the logical parent specification indicates that the concatenated key of the logical parent is to be stored as part of the segment being defined. 'V' indicates that the logical parent concatenated key should not be kept as a part of this segment. The 'P/V' subparameter affects only the size of the data portion of the segment currently being defined.

4.1.3.2 Pointer Prefix

The parameter which affects the prefix of the segment specified by the SEGM macro is the PTR parameter. This parameter specifies the pointer types that are to appear in the prefix of the segment defined by the SEGM macro. For each of the pointer types requested, 4 bytes of prefix space must be allowed for the segment. A complete specification of the values that may be used with this parameter and their meanings may be found in the IMS/VS Utilities Reference Manual.

LCHILD Pointer Parameters

The PTR parameter in the LCHILD macro specifies the number of logical child type pointers that are to appear in the prefix of the logical parent segment.

The values that may be given for PTR= are:

SNGL indicates that one 4-byte pointer is to be allocated in the prefix of the logical parent.

DBLE indicates that two 4-byte pointers are to be allocated in the prefix of the logical parent.

INDX is used to establish the relationship between a HIDAM root segment and the HIDAM primary index segment or the relationship between a target segment and the index pointer segment in a secondary index. It indicate that a 4-byte pointer is to be allocated in the prefix of the index segment in the INDEX data base.

SYMB is used with secondary indices to indicate that no pointer space is to be reserved in the prefix of the index segment. Instead, the concatenated key of the target segment is to be stored as part of the data portion of the index segment.

4.1.3.3 Parameters and macros that require VSAM

Certain parameters of the DBD generation may only be used when the Operating System access method has been specified as VSAM. These parameters are invalid if they are used for nonVSAM data bases.

COMPRTN=routine

This is a parameter of the SEGM macro and specifies the name of the segment edit/compression routine to be used when storing or accessing this segment. Any segment for which an edit/compression routine is requested by the use of this parameter must reside in a VSAM data set.

BYTES=(maximum,minimum)

This is also a parameter of the SEGM macro and is used when variable length segments are being defined in the data base. The normal format of this parameter is:

BYTES=segment size

and is used for the definition of fixed length segments. VSAM is a prerequisite for the use of variable length segments. It is invalid to specify the 'maximum/minimum' form of the BYTES parameter for any data base whose Operating System access method is not VSAM.

Note: VSAM will still use records of the same length except with GSAM.

XDFLD for other than HIDAM Root Index.

The XDFLD macro is used to establish a relationship either between the root segments of a HIDAM data base and the HIDAM primary index or between a target segment and an index pointer segment for a secondary index. If this macro is used for a secondary index then VSAM is required because secondary index INDEX data bases must be VSAM data sets.

4.1.4 DBDGEN Output Recommendations for VSAM Cluster Definition

In the output listing which results from performing DBD generation can be found a set of suggestions for certain VSAM cluster definition parameters. All of the other parameters for cluster definition are ignored which may lead the user to assume that these should be allowed to take their default values. This is contrary to one of the basic principles of VSAM data set design. The principle states that one should not blindly use the defaults but rather carefully evaluate them and specify the values explicitly in the definition. The parameters whose values are suggested by DBD generation are:

CLUSTER TYPE - indexed or nonindexed
RECORDSIZE - average and maximum
CONTROLINTERVALSIZE - applies only to the data component
KEYS - length and location (only for KSDS)

Although the NAME parameter is included in the set of suggestions, no actual data set name is recommended. The user is simply instructed to

supply an appropriate cluster name according to whatever installation conventions may be in use,

Many important parameters are not covered by these suggestions and must be considered by the data base designer. Among these omitted parameters are: FREESPACE, VOLUMES, Space Allocation, PASSWORD, SHAREOPTIONS, Index Options, and the NAME parameters for the index and data components of the cluster. These and other parameters of VSAM cluster definition are covered in the following section "VSAM Cluster Definition Parameters" on page 63.

4.2 VSAM DATA SPACE ALLOCATION

VSAM data sets can coexist on the same disk with nonVSAM data sets. Space defined on a volume for exclusive use of VSAM is called a VSAM **data space**.

In OS/VS a VSAM data space can consist of a maximum of 16 extents which need not be contiguous. A volume can contain multiple data spaces. The maximum size of a data space is **one volume**. Several data spaces on more than one volume may be treated logically as one data space (e.g. for sub-allocated multivolume data sets).

VSAM uses two different data spaces:

- The suballocatable data space
- The unique data space

A **suballocatable data space** can contain one or more data sets and a data set can occupy one or more suballocatable data spaces on one or more volumes. A suballocatable data space is created by an Access Method Services command DEFINE SPACE or DEFINE MASTERCATALOG/USERCATALOG.

All suballocatable data spaces on one volume are treated as one logical space, and it is possible to delete all empty suballocatable data spaces on a volume together. Individual suballocatable data spaces can not be deleted.

A VSAM cluster in a suballocatable data space is called a **suballocated cluster** or a **suballocated data set**.

A **unique data space** (also called **nonsuballocatable** data space) contains only one data set. This data set occupies only one nonsuballocatable data space on a volume (with up to 16 extents (OS/VS only)), but multiple volumes can be used for this data set. A unique data space is automatically built when a cluster or alternate index is defined with the UNIQUE attribute.

A VSAM cluster/data set in a nonsuballocatable space is called a **unique cluster** or **unique data set**.

Unique data spaces cannot be shared with other VSAM data sets.

VSAM data spaces cannot be shared with nonVSAM data sets.

4.3 VSAM CLUSTER DEFINITION PARAMETERS

4.3.1 Cluster Related Parameters

4.3.1.1 NAME

The name specified for a VSAM data set may contain 1 - 44 alphanumeric characters, national characters (, and %) and two special characters (hyphen and 12-0 overpunch).

Names containing more than eight characters must be segmented by periods; one to eight characters may be specified between periods.

The first character of a name or name segment must be either an alphabetic or national character.

With multiple catalogs you should take care that a data set name in one catalog is not duplicated in another catalog.

It is possible to have the same data set name in more than one catalog.

Access Method Services prevents you from cataloging two objects with the same name in the same catalog, and from altering the name of an object that its new name duplicates the name of another object in the same catalog.

In **MVS systems**, the data set name can also be used to identify the catalog where the data set is cataloged. A name segmented with periods is called a **qualified name**. The first qualifier (the part of the name up to the first period) is used to find the user catalog if no STEPCAT or JOBCAT catalog is specified (the Access Method Services manual contains a section 'Order of catalog use' for each command, where the catalog search sequence is described).

Note (MVS only): As the data set name (qualified name) is used for catalog allocation a qualified name and an unqualified name cannot exist in the same catalog if the first qualifier of the qualified name is the same as the unqualified name.

To prevent problems, it is suggested always to use qualified names in an MVS system and always use the same first qualifier per application.

This first qualifier should be identical with the catalog name defined for the user catalog, or one of its ALIASes.

4.3.2 Space Allocation (CYLINDER, TRACKS, RECORDS, VOLUMES)

4.3.2.1 Small data sets

For data sets less than one cylinder in size it is more advantageous for space utilization considerations (but not for performance) to specify the maximum number of **TRACKS** required in the primary allocation of the data component, one track for the nonimbedded sequence set index, and no secondary definition for either data or index. The allocations for this data set should be set so that only 1 index record is allocated. This is possible if only one CA is allocated for the data component (see also next section).

4.3.2.2 Multiple cylinder data sets (not multivolume)

It is usually best to calculate the number of **CYLINDERS** needed for data in a newly created data set and specify this amount in cylinders for the primary allocation of the data component. Make the secondary allocation equal or greater than one cylinder but less than the primary allocation.

VSAM calculates the data CA size based on primary and secondary allocations:

- If primary **or** secondary allocation is smaller than one cylinder, the smaller value (rounded up to full tracks in case of **RECORDS**) is used as CA size (CA-size in a UNIQUE data set is set to one cylinder).

- If primary **and** secondary allocation is larger than one cylinder, the CA size is one cylinder (maximum CA size).

Note that specifying space in tracks or cylinders make device conversion more difficult.

4.3.2.3 Multivolume data sets

When defining multivolume data sets, the following rules must be considered:

1. For suballocatable multivolume data sets a VSAM data space must exist on **all** volumes specified in the **VOLUMES** parameter.
2. For unique multivolume data sets all volumes which are specified in the **VOLUMES** parameter (except the first volume in the list), must **already be owned** by the catalog, in which the multivolume data set is to be cataloged.
3. The primary or secondary allocation value **must not** exceed the maximum capacity of one volume.

Explanation of the rules:

Rule 1:

Suballocated data sets:

Before defining the multivolume data set, a DEFINE SPACE command must have been issued for all volumes defined in the VOLUMES parameter of the multivolume data set.

When there is not enough space in a VSAM data space for a secondary allocation, the data space may be extended.

UNIQUE data sets:

Check if all volumes of the VOLUMES parameter (except the first) belong to the catalog, in which the data set is to be cataloged. If not, issue a DEFINE SPACE CANDIDATE command for all volumes not belonging to that catalog (except the first). After correct operation, define the multivolume data set.

Note: All volumes specified in the DEFINE SPACE CANDIDATE command must be mounted, since the VSAM ownership bit in the F4-DSCB is set ON and the volume timestamp is initialized.

In OS/VS systems when using recoverable catalogs also a F1-DSCB describing the one cylinder CRA is written into the VTOC of each volume when the command is executed.

Rule 2:

VSAM allocates the primary space on the first volume at DEFINE-time. When loading the data set and the primary space is filled, the secondary amount of space is allocated on the first volume as long as there is enough space available. If there is no more space for secondary allocations on the first volume, VSAM allocates **primary** space on the second volume. Then secondary space is allocated on volume 2, etc.

Generally, VSAM always allocates primary space on a new volume first and continues with secondary allocations. This is important for the last volume. When, for example the space allocation was CYL (200,50), then 200 cylinders are allocated, even when only 20 cylinders are needed. In this case, the specification CYL (50,200) would fit better.

4.3.2.4 Data space allocation and extension

The location of a VSAM data space or of a UNIQUE data set cannot be specified with OS/VS JCL. The size is specified in the DEFINE SPACE or DEFINE CLUSTER ... UNIQUE command.

When VSAM recognizes that a data space or a UNIQUE data set must be allocated or extended (secondary allocation), OS/DADSM is automatically called to allocate the requested space. A **data space** can only be extended on **one** volume. When a multivolume UNIQUE data set has to be extended to the next volume (when there is not enough space on the first one) OS/DADSM will perform the allocation.

When VSAM tries to allocate secondary space for a suballocated data set and there is not enough room in the VSAM data space, VSAM , and in sequence OS/DADSM allocates an amount of space either in data space secondary allocation size, or, if this is not enough, in the requested data set secondary allocation size. A data space, however can only allocate secondary extents if a secondary value was specified in the DEFINE SPACE command.

4.3.2.5 Cluster Type

The specification of the keyword **INDEXED** (the default) is used to define a KSDS cluster, while **NONINDEXED** must be specified to define an ESDS cluster.

4.3.2.6 Password Specification

An expanded password protection facility is supported for VSAM. Optionally, passwords can be defined for clusters, cluster components (data component and index component), and VSAM catalogs. VSAM passwords are kept in VSAM catalog entries only. The password is supplied by IMS/VS in using the password as DBD name.

If password protection is indicated for a VSAM data set, the operator will be prompted if a **BATCH** job is executed and if **PASSWORD=NO** has been specified in the DBDGEN. If **PASSWORD=YES** has been specified and the DBD name is not the same as the Password, a VSAM OPEN error will occur and the data base will not be opened.

The number of operator retries allowed (from 0 to 7) can be specified with the ATTEMPTS parameter (default is 2) in the DEFINE command.

Four levels of password protection are provided:

- **LEVEL 1: MASTER PASSWORD**

  Full access, which allows access to a data set, and its catalog entry. Any operation (read, add, update, delete) can be performed on the data set and its catalog entry.

  The master password is required for altering (ALTER) and deleting (DELETE) VSAM data sets.

- **LEVEL 2: CONTROL INTERVAL PASSWORD (for special usage)**

  Control interval access, which allows the user to read and write entire control intervals using the control interval interface. All read, write, and update operations can be performed at the logical record level as well. This facility is not provided for general use and should be reserved for system programmer use only.

- **LEVEL 3: UPDATE PASSWORD**

  Update access, which allows logical records to be retrieved, updated, deleted, or added.

  For catalog operations the following applies:

  - **VS1, SVS:** The Update password is required for defining VSAM data sets and nonVSAM data sets. No passwords are required for altering (ALTER) and deleting (DELETE) a nonVSAM data set.

  - **MVS:** The Update password is required for defining (DEFINE) VSAM and nonVSAM data sets and for altering (ALTER) and deleting (DELETE) nonVSAM data sets (including GDGs and ALIASes).

- **LEVEL 4: READ PASSWORD**

  Read access, which allows access to a data set for read operations only. Read access to the catalog entries of the data set (except password information) is permitted also. No writing is allowed.

  A password can be defined for a given VSAM data set for each level protection: master password, control interval access password, read-write-add-delete password, and read-only password. When multiple passwords are defined for a data set, the password given when the data set is opened establishes the level of protection to be in effect for this OPEN.

  Authorization to process a VSAM data set can be supplemented by a user-written **security authorization routine.**

  If supplied, such a routine must reside in the library SYS1.LINKLIB or in a library to be located by the system (e.g. in a JOBLIB).

  The authorization routine is entered during OPEN processing after password verification has been performed by VSAM, unless the master

access password was specified. A user security authorization record of up to 255 bytes maximum can also be added to the catalog entry for the data set. This record can supply data to the user-written security authorization routine during its processing. For further information see the **OS/VS VSAM PROGRAMMER'S GUIDE, GC26-3838.**

The VSAM catalog must be password protected to support password protection for cluster passwords.

If passwords are not specified on **all levels**, the highest level password is propagated to the higher levels.

Passwords for catalogs are strongly recommended to prevent unauthorized deletion of VSAM objects (e.g. clusters, data spaces and catalogs).

In OS/VS systems the VSAM master catalog master password is required for volume cleanup using the Access Method Services command ALTER REMOVEVOLUMES and other severe commands. Therefore, it is very important to have **at least** the VSAM master catalog password protected.

4.3.2.7 SHAREOPTIONS

A VSAM data set can be accessed concurrently by two or more subtasks within the same partition and two or more job steps (partitions) DISP=SHR must be specified for the VSAM data set by each job step. Both types of sharing can be used for a VSAM data set at the same time. The type of data set sharing permitted for two or more partitions is controlled by using the SHAREOPTIONS parameter of the DEFINE command when the VSAM data set is defined. **Note that there is no intergrity support in IMS/VS for sharing data bases between jobs other than ONLINE.**

The format of the **SHAREOPTIONS** command is as follows:

SHAREOPTIONS (cross-region cross-system)

VSAM sharing and **SHAREOPTIONS** are described in detail in section "VSAM Sharing" on page 83.

4.3.2.8 UNIQUE/SUBALLOCATION

As described in section "VSAM Data Space Allocation" on page 62 VSAM uses two different types of data spaces/data sets in terms of allocation.

The **SUBALLOCATION** parameter specifies that a data set shares a predefined VSAM data space with other VSAM data sets.

VSAM does all the space management and no JCL specification for the location of the data set must be given.

VSAM has been designed to use SUBALLOCATION.

The advantage (if using suballocated data sets) is that the 16 extent restriction per volume does not apply and therefore the disk space can be used more flexibly. A disadvantage is that the space allocated via DEFINE SPACE is taken away from DADSM free space on the volume and therefore cannot be used to satisfy nonVSAM allocations even if it has not yet been used by a cluster.

The **UNIQUE** parameter requires a separate allocation for the data set. One F1-DSCB (Label) (or two for a KSDS) are written into the VTOC for the UNIQUE data set. The name in the F1 DSCB is the **data component name** (and for a KSDS additionally the **index component name**).

The **cluster name** specified in the DEFINE command is **not** used in the VTOC. Therefore it is recommended to specify also a name for the data component (and in case of a KSDS, also a name for the index component), since otherwise VSAM would generate its own 44-Byte name which is meaningless to the user when listing the VTOC or the catalog (LISTCAT).

UNIQUE data sets should be restricted for special applications.

Advantages of SUBALLOCATION:

1. Since it is the default, there is no need to specify it on an Access Method Services DEFINE command.

2. If the suballocated cluster is on a different volume from the owning catalog, it is possible to DELETE the cluster without the cluster's volume being mounted (if a nonrecoverable catalog is used). This is because all the information which needs to be changed by the DELETE is in the catalog.

3. The REUSE attribute, which allows the VSAM user to reload VSAM clusters without the need for DELETE/DEFINE, is only supported for SUBALLOCATION clusters.

4. VSAM space management tries harder than DADSM to avoid fragmentation. When looking for a place to put a suballocated cluster, VSAM will try to find the smallest contiguous amount of free space which satisfies the cluster's primary allocation requirement, even if a more suitable allocation is possible elsewhere on the volume.

5. If a volume contains many unique data spaces unexpected IMS/VS wait times may occur when a BMP needs to extend a unique data set. This EOV request must run in the control region and BMP is waiting for EOV to finish. The extension of a unique data space is internally equivalent to a DEFINE SPACE. The more spaces exist on a volume, the longer DEFINE SPACE is going to run. During this entire period the control region is unavailable. As each BMP requires control region services (ISWITCH) it will wait until the control region is available. Eventually all BMPs will be waiting for the control region.

Advantages of UNIQUE:

1. There is only one VSAM cluster component per VTOC entry, with the possibility of making VTOC names the same as cluster component names. The organization of handling DASD space is the same as for nonVSAM data sets. VTOC related 'emergency' operations, such as SCRATCH and zapping of password bits, can be isolated to one VSAM cluster component.

2. On a volume with a large amount of VSAM space and a large number of clusters, use of UNIQUE allocation may result in better performance for DEFINE. This is because VSAM space management expends a lot more energy in making the best possible fit, for the DEFINE request. This may result in high CPU times and many I/O operations to the catalog for suballocated DEFINEs.

4.3.2.9 SPEED/RECOVERY

When a VSAM data set is loaded, VSAM does or does not preformat control areas, depending on the attribute specified when the data set is defined, **RECOVERY** or **SPEED**, respectively.

When **RECOVERY** (the default) is specified, during loading VSAM preformats each control area immediately before loading any records into it. Preformatting for a key-sequenced data set consists of putting the appropriate control information in each control interval of that control area and an end-of-file indication in the first control interval of the next control area. All zeros in the control interval definition field indicates end of file or end of key range for a key-sequenced data set.

For an entry-sequenced or relative record data set, control information and an end-of-file indication are placed in each control interval of the control area during preformatting.

The **RECOVERY** option (**not suggested**) ensures that if an error occurs, which prevents further processing while a control area is being loaded, the previously loaded control areas are not lost. Loading can resume from the first or only end-of-file indicator. Preformatting in **RECOVERY** mode is always done when records are added to an existing VSAM data set (even if **SPEED** was specified). Thus, after the first time the data set is CLOSEd, it is always processed in RECOVERY mode.

When **SPEED** is specified (suggested), records are loaded (i.e. between **first** OPEN and CLOSE) without preformatting each control area before loading and the end-of-file indicator is not written until the data set is closed. When this option is chosen, loading proceeds more rapidly, but if an error that prevents further processing occurs, all the records loaded up to that point may be lost and loading would have to resume at the beginning of the data set.

The following should be considered when using RECOVERY (KSDS only):

- Up to now when an empty data set is loaded, the 'high used RBA' is only updated if the data set has been closed (by the user) after at least one record written into it, or if a second extent has to be allocated.

- Since VERIFY does **not** work for a data set with a 'high used RBA = 0', which usually indicates an empty data set, loading could not be resumed, even when **RECOVERY** is specified.

- If the user-written load program had issued a CLOSE while loading the data set (which REPRO does **not** do), VERIFY may be used after a system malfunction to adjust the 'high used RBA' and the loading may be resumed (the user has to check which record was entered last and he has to change the load program to resume loading with the next record).

- It is suggested that one always specify **SPEED** which is **not** the default.

4.3.2.10 IMS/VS Considerations using SPEED or RECOVERY

There are different considerations for the use of SPEED or RECOVERY depending on the type of data base being used.

For the Hierarchical Direct organizations, HDAM and HIDAM, it does not matter which option is specified. This is because IMS/VS OPENs these data bases, writes a control record and then CLOSEs the data base again. The data bases are then reOPENed to load the data base records. This procedure causes the data sets comprising the data bases to be processed in RECOVERY mode during the load process regardless of the option chosen because they have already been CLOSEd once.

HDAM and HIDAM data bases will only use SPEED, if that option has been specified, during the period when the initial control record is written.

The situation is different for the KSDS part of HISAM and SHISAM data bases. No special control records are written into these data bases so the selected option will be in effect until the data base is CLOSEd after loading has been completed. SPEED is recommended in order to reduce the time taken by the load. Restart is not possible (see RECOVERY considerations above).

The ESDS part of a HISAM data base is treated as a HDAM data base. The case of an INDEX (primary or secondary) is also handled the same way with a dummy record being inserted first. After CLOSE and a following OPEN this record is erased. Therefore, no matter which option is selected, the INDEX will be loaded using the RECOVERY option.

It should be noted that once the initial load of the data bases has been completed, they are always processed in RECOVERY mode.

4.3.3 DATA Component Related Parameters

4.3.3.1 NAME

It is advisable to define a separate name for the DATA component of the cluster. If no name is specified by the user, VSAM will assign a 44-byte name which consists of timestamp fields and other data.

If, for example, a cluster is named 'CUSTOMER.NAMES' the DATA component could be named 'CUSTOMER.NAMES.D'. This allows easy identification of the components of the cluster when analyzing a list of the catalog.

Sometimes it might be necessary to change parameters of an existing data set using the Access Method Services command ALTER. If, for example, a different FREESPACE value is to be assigned, the DATA component name (not the cluster name) must be specified.

The naming conventions are described in section "NAME" on page 63.

4.3.3.2 CONTROLINTERVALSIZE

The control interval size must be a multiple of 512 bytes unless the control interval size is greater than 8192, then the multiple is 2048. The maximum size of a control interval is 32,768 bytes.

DBDGEN output includes a suggestion for the control interval size, which should be carefully evaluated before it is used.

4.3.3.3 Space Allocation at DATA Level

The space allocation for a data set can be specified on the cluster level, or separate on the component level. In general, it is advisable to specify the calculated data set size on the cluster level.

Space allocation considerations are described in section "Space Allocation (CYLINDER, TRACKS, RECORDS, VOLUMES)" on page 64.

4.3.3.4 FREESPACE

A KSDS can be specified to reserve space to be held free when loading data.

The specified amount is only held free when sequential insertions such as 'loading' are performed.

Two values can be specified:

FREESPACE(CI-percent CA-percent)

'CI-percent' specifies the amount of space to be held free per control interval (CI).

The way VSAM treats this free space definition is: specified percentage times actual CI size (rounded down to a full byte). If the amount of free space is less than the size of one record the value is increased to the size of one record.

Example:

Definition:

Control Interval Size = 1024 bytes
Free space = 15 % free CI space (FREESPACE (15 0)

VSAM calculation:

1024 * 15 % = 153,6 = 153 bytes minimum free space (excluding control fields).

'CA-percent' specifies the amount of control intervals to be held free per control area (CA). If the amount of free space is less than the size of one CI the value is increased to the size of one CI.

If the maximum value (100%) is specified for both **'CI-percent'** and **'CA-percent'**, each control interval will contain one record and each control area will contain one used control interval.

4.3.3.5 PASSWORDs for DATA Component

Passwords on the DATA component level have no effect for IMS/VS access. They may be used as an additional protection against unauthorized access from other programs.

4.3.3.6 KEYS

The format of the KEYS parameter is:

KEYS (length offset | 64 0)

This parameter specifies information about the key field of record in a KSDS cluster.

The **'length'** can be from 1 through 255 bytes (default=64).

The **'offset'** specifies the displacement of the key field (in bytes) from the beginning of the record (default=0).

DBDGEN output includes a suggestion how to specify the KEYS parameter.

4.3.3.7 RECORDSIZE

There is no special parameter in VSAM to define that fixed length records or variable length records are to be used, only the average and maximum length of a record may be specified.

The format of RECORDSIZE is:

RECORDSIZE (average length maximum length)

If fixed length records with a length of 80 bytes are to be used, the definition would be RECORDSIZE (80 80).

The definition of **RECORDSIZE (60 80)** would, however, also enable the user to use fixed length records with 80 bytes, because the **'average length'** parameter is only used by VSAM to calculate the minimum control interval size for the data component (or to check the size the user has defined).

The **'maximum length'** parameter is a limiting size for VSAM. Records with a larger size will force an error message, but if **RECORDSIZE (100 100)** has been specified and only records with a length of 80 bytes are used, neither error messages are written nor is space wasted on disk (except for an RRDS data set whose slots are preformatted depending on the **'maximum length'** parameter), as VSAM checks the actual length of the record and stores this information in the RDF (Record Definition Field).

Normally both values **average length** and **maximum length** are identical, except for GSAM which may use variable length records.

DBDGEN output includes a suggestion for the RECORDSIZE parameter.

4.3.4 Index Component Related Parameters

4.3.4.1 NAME

It is advisable to define a separate name for the INDEX component of the KSDS cluster. If no name is specified by the user, VSAM will assign a 44-byte name which consists of timestamp fields and other data.

If, for example, a cluster is named 'CUSTOMER.NAMES' the INDEX component could be named 'CUSTOMER.NAMES.I'. This allows to easy identify components of the cluster when analyzing a list of the catalog.

The naming conventions are described in section "NAME" on page 63.

4.3.4.2 CONTROLINTERVALSIZE

The INDEX control interval sizes are:

512, 1024, 2048, and 4096 bytes

It might be advisable to specify a different INDEX control interval size then the DATA control interval size. For further details see section "VSAM BUFFER POOL Definition Parameters" on page 77.

DBDGEN output includes a suggestion for the control interval size.

4.3.4.3 Space Allocation at INDEX Level

The space allocation for a data set can be specified on the cluster level, or separate on the component level. In general, it is advisable to specify the calculated data set size on the cluster level. VSAM will then take the necessary amount from this space for the INDEX component.

Space allocation considerations are described in section "Space Allocation (CYLINDER, TRACKS, RECORDS, VOLUMES)" on page 64.

4.3.4.4 PASSWORDs for INDEX Component

Passwords on the INDEX component level have no effect for IMS/VS access. They may be used as an additional protection against unauthorized access from other programs.

4.3.4.5 IMBED, REPLICATE

When a key-sequenced data set is processed sequentially, the sequence set index level is used to indicate the order in which control intervals are to be accessed. To improve performance during sequential processing, the index sequence set can be separated from the rest of the index component (index set) and stored with the logical records in the data component (IMBED). When this option is chosen, the index records for a control area are placed on the first track of the control area so that both index and logical records can be accessed without moving the disk arm (similar to the location of the track index within the prime area in an ISAM data set).

When the index sequence set is stored within the data component (IMBED), sequence set records are also replicated. That is, each sequence set index record is allocated one track at the beginning of the control area. The index record is duplicated on the track as many times as it will fit. This technique significantly minimizes the rotational delay involved in arriving at the beginning of an index record. If there is only one control area in a cylinder, index sequence set records will be replicated beginning with track 0. If there are two control areas in a cylinder the initial track of the first area will contain replicated index records for the first control area, while the initial track of the second area will contain replicated index records for the second control area.

Index set records, like sequence set records, contain compressed index entries. The index entries in each level of the index set point to index records of the next lower index level. An index entry within the index set contains a pointer to an index record, the highest key in that index record, and control information.

Index set levels can also be replicated (REPLICATE). When this option is chosen, one track is required for each index record in the entire index set. An index record is duplicated on its assigned track as many times as it will fit.

With REPLICATE performance is even more improved in direct processing rather than in sequential processing. In sequential processing an index record is read once for a CA; in direct processing each data record read operation may require an extra read operation for the index record.

The index set may not be replicated when the index set and the sequence set of the primary index are physically separate (sequence set stored with logical records). However, when the index set and the sequence set are stored together, both are replicated or neither is replicated.

The following combinations are possible:

- **NOIMBED NOREPLICATE:** All index levels are stored together and are not replicated.

- **NOIMBED REPLICATE:** All index levels are stored together, each index record occupies its own track and is repeated on this track as often as possible.

- **IMBED NOREPLICATE:** The index sequence set level is stored with the data component. Each index record in the sequence set is stored with its data control area, occupies its own track, and is repeated on this track as often as possible. **In most cases this is the best choice.** The higher index levels are not repeated on a track.

- **IMBED REPLICATE:** The index sequence set level is stored with the data component. Each index record in the sequence set is stored with its data control area. Each sequence set record occupies its own track, and is repeated on this track as often as possible.

  To improve performance during processing, each sequence set index record may be stored with the CA it points to. This generally eliminates disk-arm movement because it is not necessary to do separate seeks to locate both the sequence index record and the data record (if the arm has not been moved by other activities between index and data reference).

A data CI can never reside on the same physical track as the index CI. Most of the index (index sequence set records and all higher level index records) can reside in virtual storage if enough buffer storage is specified by the user, but VSAM does not preload index buffer(s). Depending on the type of access index buffers are assigned and used differently.

Only one sequence set record can be resident in a buffer with NSR. This should be considered, because IMS/VS is not always using Shared Resources.

4.4 VSAM BUFFER POOL DEFINITION PARAMETERS

The VSAM buffer pools are defined by a set of user supplied statements which are contained in a special data set. For the IMS/VS batch (DB only) environment this data set is identified in the JCL by the DD name, **DFSVSAMP**. In the online (DB/DC) environment, the data set must be a member of the IMS/VS Procedure Library, IMSVS.PROCLIB. The member name is **DFSVSMxx**, where the suffix ('xx') is specified as the value of the VSPEC parameter on the JCL EXEC statement.

There are two types of statements used in this data set. One is the buffer subpool definition statement which is explained in section "Subpool Definition" on page 78 and the other is the **OPTIONS** statement which is explained in section "OPTIONS Statement" on page 80.

4.4.1 Subpool Definition

The subpool definition statement follows the **DFSVSAMP DD-statement**. It may start in any column and has the following format:

buffer size,number of buffers

The **'buffer size'** parameter may be specified as a 3 to 5 digit number and specifies the size of the buffers in this subpool. Permissible values for buffer size are 512, 1024, 2048, 4096, 8192, 12288, 16384, 20480, 24576, 28672 and 32768 (note that these buffer sizes do not completely correspond to all of the available CI sizes).

The **'number of buffers'** parameter is a 1 to 3 digit number and specifies how many buffers of this size are to be present in this subpool.

The maximum value that may be specified is 255.

The minimum value depends on the environment in which the data bases are being processed. For batch the minimum is 3 and for online the minimum is (2 * the number of dependent regions[14])+ 1. If the 'number of buffers' given is less than the minimum required, it is increased to the minimum and a warning message is issued.

A number of subpool definition statements may be included in the data set. Each statement defines a separate subpool of buffers except when two or more statements specify the same buffer size. In this case, the number of buffers specified on each statement are summed and a single subpool with the total number of buffers is built.

4.4.1.1 Selection of Buffer Sizes

There is no facility available to allow the user to assign a data set to a particular buffer subpool. The determination of which of the defined subpools a data set will use is done at IMS/VS initialization on a 'best fit' basis. This means that the data set component will be allocated to that subpool whose buffer size is the smallest which is able to contain the component control intervals. Because the buffer sizes do not exactly correspond to the CI sizes, in many cases a buffer larger than the CI will be used with the extra portion of the buffer being wasted.

---

[14] The number of dependent regions is given as the first subparameter of the **MAXREGN** parameter of the **IMSCTRL** macro in the IMS/VS system definition. For further information about the IMSCTRL macro and MAXREGN parameter refer to the IMS/VS Installation Guide.

The only form of control over buffer subpool usage that can be exercised by the user is in the selection of CI sizes when the clusters are defined and the definition of appropriately sized buffer subpools. Whenever possible, CI sizes for the VSAM clusters should be chosen so that the following guidelines are met:

1. Index and Data components should be placed in different subpools in order to avoid contention for the same set of buffers. This will also help in allowing the high level index structures to be kept in storage and thereby improve performance.

2. Only one frequently accessed data set should be allocated to each subpool. This will prevent competition between the high usage data sets which may lead to frequent I/O operations and slow system response.

3. Activity should be balanced across the subpools by grouping data sets so that the access requirements of all groups are approximately equal. This will prevent a particular subpool from becoming a bottle neck in the system.

4. Control interval sizes should match buffer sizes as closely as possible in order to avoid wasted space in the buffers.

4.4.1.2 Choice of Number of Buffers

The number of buffers to be allocated in each subpool depends to a great extent on which data sets are to be assigned to the subpool.

When the subpool is being used by index data sets, a large number of buffers is advantageous for random processing but relatively useless when the access is primarily sequential. In the IMS/VS online (DB/DC) environment, it is most likely that random access will be prevalent so a large number of index buffers is desirable.

The number of buffers required for each of the index data sets sharing the subpool is the number of records (CIs) in the high level portion of the index (sequence set records are excluded) plus the number of dependent regions as given in the MAXREGN parameter (in the batch environment of IMS/VS, there is only one dependent region).

The number of buffers in the subpool should be the total of the requirements of the data sets sharing the subpool plus 1. This allocation will allow the high level index structures to be resident in the buffers and

will supply one sequence set buffer per dependent region for each data set. Since VSAM performs full look-aside across the buffers, additional buffers may be useful in that they will let more sequence set records to be maintained in current status in the subpool.

If the subpool is being used for data CIs, then the number of buffers need not be large. VSAM does not perform 'read ahead' for sequential accessing so that to have many data buffers is only useful in aiding look-aside or to improve control area splitting. Since genuinely random processing is unlikely to reuse the content of a buffer very frequently and CA splits will not occur often, extra data buffers do not materially aid performance.

The minimum number of buffers to be placed in subpools shared by the data components of VSAM clusters depends to a certain extent on the type of data base to which the data sets belong. For HDAM and HIDAM data bases, 2 buffers are needed for each dependent region. HISAM data bases require 3 buffers per dependent region. Each data base also needs 1 extra buffer to be available for CI and CA splitting. The total number of buffers in the subpool is sum of the requirements for all of the data sets which share the subpool.

4.4.2 OPTIONS Statement

The OPTIONS statement and all of its parameters are optional. The statement must begin in position 1 and may not be continued on a subsequent line. However, several OPTIONS statements may be given and each may specify a different set of parameters. If a parameter is repeated, the last occurrence will determine the value to be used.

The parameters of the OPTIONS statement are:

```
BGWRT=(YES,34)|NO|(YES,nn),
INSERT=SKP|SEQ,
VSAMFIX=BFR|IOB|(BFR,IOB),
VSAMPLS=GLBL|LOCL,
BHTRACE=nnn,
DUMP=NO|YES,
LTWA=NO|YES.
```

Figure 35. Parameters of the OPTIONS statement: The default values for the parameters are shown underscored.

Only the first four parameters, BGWRT, INSERT, VSAMFIX and VSAMPLS, have an effect in terms of VSAM. The other three parameters do not fall within the scope of this document and are therefore not discussed. For information on BHTRACE, DUMP and LTWA refer to the **IMS/VS INSTALLATION GUIDE, SH20-9081.**

Note: As a general rule, LTWA (Log Tape Write Ahead) should **always** be specified as **YES** for the online environment of IMS.

The **BGWRT** parameter determines whether (YES) or not (NO) the **Background Write** facility is to be used. Background Write is a low priority task in IMS/VS which inspects a certain percentage of the buffers in each subpool to determine if they are eligible to be written out to the data base. The percentage of the buffers to be checked is given as the second subparameter, 'nn', when YES is specified and may be any value from 10 to 99. The default is YES,34.

Buffers are checked on a 'least recently referenced' basis and are considered to be candidates for writing to the data base if their contents have been modified by the application programs. Use of the Background Write facility can aid performance because it frees buffers for reuse before they are needed. Otherwise, when the subpool is full, IMS/VS would have to wait until a buffer could be written out before being able to read in a new control interval to satisfy a request. For further information on the operation of Background Write, refer to the section DL/I Data Base Buffering Facilities in the **IMS/VS SYSTEM PROGRAMMING REFERENCE MANUAL, SH20-9027.**

The **INSERT** parameter determines the insert mode that will be used for adding new KSDS records in the data bases. The operation and effect of the modes of insertion was discussed earlier in "Insert Strategies Effect on Splitting" on page 32. Specification of **SEQ** will cause the buffer handler to use VSAM sequential mode insertion and is useful when several groups of root segments are to be inserted in key sequence. If **SKP** is specified or allowed to default, then the buffer handler will use VSAM direct mode insertion. Direct mode insertion is the standard method of handling inserts in VSAM. SKP also causes inserts to be performed in skip-sequential mode as explained in section "Skip-sequential Insertion" on page 99.

The **VSAMFIX** parameter is used to request that the VSAM resource pools be page-fixed in main storage. **BFR** specifies that the buffer subpools and their associated control blocks should be fixed. **IOB** specifies that the I/O related blocks, such as RPLs and PLHs, should be fixed. This parameter has no default and if it is not specified then nothing is fixed and the buffer subpools and I/O blocks are pageable. Requesting page-fixing will improve performance in VSAM input and output since no page exceptions will have to be processed but it may reduce the available working storage to the point that the rest of the system begins 'thrashing'.

The VSAMPLS parameter determines where the VSAM shared resource pools are to be built. If LOCL is specified, then the Local Shared Resource (LSR) facility of VSAM is used and the VSAM control blocks and buffer subpools are built in the IMS/VS control region. GLBL applies only to OS/VS MVS operating systems and specifies that the Global Shared Resource (GSR) facility of VSAM is to be used with the control blocks and buffer subpools being built in the Common Service Area (CSA). If GLBL is specified or allowed to default for a VS1 operating system, it is treated as though LOCL had been specified.

Only one IMS/VS system at a time may use GLBL as OS/VS VSAM will only allow a single global (GSR) pool for each storage key (0-7) in a processor or MP system. If it is necessary to have two separate IMS/VS systems running in the same processor or MP, then one of them must be using a local (LSR) pool and have LOCL specified.

5.1 SHARING OF DATA IN IMS/VSAM ENVIRONMENT

There are two aspects to the sharing of VSAM data bases in an IMS/VS environment. The first is the sharing capabilities and restrictions native to VSAM and the second is the considerations and operations of IMS/VS with regard to multiple users of a data base. These two aspects are treated separately in the following discussion. The interaction of IMS/VS with VSAM sharing capabilities is covered in the second section "Sharing of VSAM Data Bases" on page 87.

5.1.1 VSAM Sharing

Native VSAM sharing is covered first in that it forms the basis for the sharing of data bases which are implemented by VSAM data sets. The facilities of VSAM for sharing data sets can be divided into two categories: those which support sharing within a task or region and those which support sharing between regions or systems.

5.1.1.1 Sharing Across Tasks Within a Region

VSAM's ability to allow multiple subtasks in the same region or partition to share data sets is called by a number of names. It is known variously as:

- subtask sharing
- DDNAME sharing
- data set name (DSN) sharing

The facility for sharing within a region provides a certain level of data integrity by ensuring that a single control block structure is used to access the data set. It also conserves resources by avoiding the need to have multiple copies of the control block structure. This type of sharing may occur simultaneously with cross-region/system sharing. All OPENs to the shared data set must be issued from the same partition or address space and must reference the same VSAM cluster.

When using this level of sharing, VSAM will maintain write integrity for the data set but will **not** maintain read integrity. Write integrity is maintained by 'locking' a control interval which is accessed for update. Once a control interval has been 'locked' by an update request, any attempt to access the same control interval for update purposes by another request will result in a VSAM logical error. This effectively prevents any

overlap or dual update problems with the data set. A nonupdate request to a control interval that has been 'locked' will not be prevented however. Thus, in certain circumstances, it is possible that a nonupdate request may get a down-level version of the control interval and be working with data that is not current.

The only way in which to achieve read integrity for the data set at this level of sharing is to make all requests in update mode.

5.1.1.2 Sharing Across Regions or Systems

When multiple control block structures are used to access the same VSAM data set, whether these structures are in the same region or not, the cross-region/system sharing facilities of VSAM are used. This level of sharing is governed by the **SHAREOPTIONS** specified for the VSAM data sets in the cluster definition. SHAREOPTIONS do not have any effect on the sharing of a data set across tasks in the same partition or region.

The format of the **SHAREOPTIONS** parameter in the cluster definition is:

SHAREOPTIONS(cross-region cross-system).

'cross-region' applies to sharing of the data set between multiple regions or partitions within the same operating system. SHAREOPTIONS apply when the data set is shared between multiple operating systems.

Values that may be specified are 1, 2, 3 or 4 (1 and 2 for **'cross-region'** only). The meanings of these four values are explained below.

SHAREOPTION 1:

This option is also known as **Full Integrity** since it maintains both read and write integrity. With **SHAREOPTION 1**, the data set may be OPENed by one region or partition for update purposes **OR** by any number of regions or partitions for read-only access. VSAM checks at OPEN time to ensure either that the data set is not open for some other region or partition if this OPEN is for update or that no user has the data set open for update if this OPEN is for read-only access. An invalid second OPEN request (e.g. OPEN for Write) would be rejected with an OPEN return code.

The use of this SHAREOPTION ensures that the data being used is always current and that there is no possibility of multiple update jobs coming into conflict with one another.

This value may only be specified for cross-region sharing and is **not** allowed for cross-system sharing. It is the default value for the cross-region subparameter.

SHAREOPTION 2:

This option supplies only **Write Integrity**. It allows the data set to be OPENed by one region or partition for updating AND by several other regions or partitions for read-only access. The use of this option prevents update conflicts and permits updating to be performed concurrently with enquiry-mode processing. No attempt is made to maintain read integrity with SHAREOPTION 2. It is therefore possible for the enquiry jobs to be working with down-level versions of the data.

Like **SHAREOPTION 1**, this option may only be specified for cross-region sharing and is invalid for cross-system sharing.

SHAREOPTION 3:

This option does not maintain any form of integrity and is known as the **No Integrity** option. It will permit multiple OPENs of the data set for either update or enquiry and performs no checking at OPEN time. When **SHAREOPTION 3:** is used, it is the responsibility of the user to implement either programming or operational procedures to protect the data set against update conflicts.

This option may be specified for either cross-region or cross-system sharing. It is the default value for the cross-system sharing subparameter.

SHAREOPTION 4:

This option, like SHAREOPTION 3, does not directly provide any integrity but there is a limited amount of assistance from VSAM to help the user in maintaining integrity. As with SHAREOPTION 3, when SHAREOPTION 4 is specified, multiple update and enquiry jobs are allowed to access the data set simultaneously and it is the user's responsibility to ensure that integrity is maintained.

The assistance provided by VSAM to help the user in maintaining integrity has two aspects:

1. All data buffers and sequence set buffers are refreshed for every direct (random) request in order to try to return the most current form of the data. Buffer refresh is **not** performed for sequential requests and it is possible that a sequential request could obtain a down-level version of the data.

2. There are certain restrictions placed on the processing of the data set which are intended to prevent any change in the end-of-file pointers that VSAM maintains in the catalog. These restrictions are:

   - No control area splits may take place in the data set.
   - No new extents may be added to the data set.
   - No new records may be added to an ESDS.
   - For a KSDS, the control interval which contains the highest key in the data set may not be split.

SHAREOPTION 4 may be specified for either the cross-region or the cross-system subparameter.

Note: SHAREOPTION 4 and **VSAM deferred write** are incompatible. IMS/VS uses VSAM deferred write for Shared Resources, therefore, IMS/VS will not OPEN a data set with SHAREOPTION 4. **SHAREOPTION 4 should not be specified for VSAM data sets which are components of an IMS/VS data base.**

5.1.1.3 Other Factors Affecting Sharing

In addition to the SHAREOPTIONS specified in the definition of the VSAM cluster, there are a number of other factors which will affect the sharing capability of VSAM data sets:

1. The **DISP parameter** of the JCL DD statement used to reference the data set. If DISP=OLD is specified on the DD statement, then the data set is treated as though SHAREOPTIONS(1,3) were defined for the cluster regardless of what the actual definition of the SHAREOPTIONS may be.

   This override does not change the defined SHAREOPTIONS but, rather, merely causes the data set to be OPENed in accordance with the restrictions of SHAREOPTIONS (1,3). The reason for this effect is that, when DISP=OLD is used, the Scheduler ENQs on the data set thereby ensuring that only this job may access the data set. It should be noted that DISP=OLD will not prevent a job in another CPU from gaining access to the data set unless JES3 complex-wide control is in effect.

2. The **processing mode of the data set.** When the data set is being processed in create mode (opened empty or opened RESET for reuseable data sets and being loaded with records), then it is treated as though SHAREOPTIONS(1,3) were specified.

   Like DISP=OLD, create mode does not reset the SHAREOPTIONS. This effect is due to the fact that it is assumed that the job which is performing the initial load of the data set should have exclusive control until the load has been completed.

3. The **operating system generation parameters** specified for the device(s) on which the data set resides.

   If a device on which the data set resides has not been generated as SHARED (or SHAREDUP, under MVS), then a cross-system SHAREOPTION of 4 is treated as a cross-system SHAREOPTION 3. This factor has very little effect on IMS/VS use of VSAM data sets but is mentioned here for the sake of completeness.

5.1.2 Sharing of VSAM Data Bases

In the following section IMS/VS handling of VSAM data sets and the way in which this affects sharing is covered. There are two major areas of interaction between IMS/VS and VSAM SHAREOPTIONS. One of these is the way in which IMS/VS opens the VSAM data set and the other is the relation of batch and online IMS/VS programs.

5.1.2.1 IMS/VS Open Considerations

IMS/VS always opens VSAM data sets for **update processing**. This is true whether IMS/VS is running in the online environment (DB/DC) or as a pure batch job (DL/I only). IMS/VS does not have the capability to inspect the processing options[15] for the program(s) which will access the data base in order to determine their intentions before opening the data base. It therefore always performs OPEN for update so that it may process any update requests that may be issued by the program(s).

This action by IMS/VS has the result that there is no effective difference between cross-region SHAREOPTIONS 1 and 2 when the VSAM data set is processed as part of an IMS/VS data base. Since IMS/VS will have opened the data set for update, no other program will be able to gain access to the data set if it has been defined with SHAREOPTION 1 or 2. Conversely, IMS/VS will be unable to OPEN a data set which has SHAREOPTION 1 or 2 if the data set is in use by some other program. One advantage to this philosophy of 'OPEN for update regardless' is that it will maintain full integrity for the data set even if the users have indicated that they are prepared to accept down-level data when in enquiry mode by specifying SHAREOPTION 2.

If it is desired that the VSAM data sets be shareable between IMS/VS and some other program, it is necessary to define the cluster with cross-region SHAREOPTION 3. However, this may result in problems if proper programming or operational procedures have not been implemented to protect the data set against update conflicts or improper use as there is no protection supplied by VSAM.

It is not possible to make use of cross-region SHAREOPTION 4 in this case because IMS/VS does not support SHAREOPTION 4 for LSR or GSR.

---

[15] The processing options are defined through the PROCOPT parameter of the PCB or SENSEG macros used in PSB generation. For further information on the PROCOPT parameter refer to the IMS/VS UTILITIES REFERENCE MANUAL, SH20-9029.

5.1.2.2 Batch/Online Sharing Capabilities

In order to understand the sharing process as it relates to IMS/VS data bases it is necessary to be aware of the differences in data set handling in the two IMS/VS environments, Batch (DB only) and Online (DB/DC). A brief discussion of the two environments follows.

In the **Batch mode** of operation each program runs as a separate job in its own region or address space. All control blocks used to access the data sets which comprise the data bases are located in the program's partition and are not available outside of this area. This means that, if an attempt is made to access a data base which is already in use by another program, a different control block structure must be used and the VSAM SHAREOPTIONS therefore come into effect.

As described above in "IMS/VS Open Considerations" on page 87, SHAREOPTION 1 or 2 will prevent the sharing of data sets between jobs in the DB-only environment. If sharing is necessary or desirable then the VSAM clusters must be defined with a cross-region SHAREOPTION of 3 or 4 and it will be the user's responsibility to implement procedures to avoid conflicts.

The activity of the **Online IMS/VS** system is supervised by the Control Region. In the online (DB/DC) environment of IMS, there are two different types of application programs. These are the Message Processing Programs (MPP) and the Batch Message Programs (BMP). The major difference between these two types of program is that a BMP may access normal system data sets while an MPP cannot. Access to the online data bases from an application program of either type is performed directly from the application programs. There is one exception though: when a user has specified that VSAM Local Shared Resources are to be used instead of VSAM Global Shared Resources. In this case all I/Os will be performed from the Control Region. This options is available to MVS users only.

The control blocks needed to access the data bases are kept in the address space of the Control Region and all OPENs are issued from the Control Region. Thus, all application programs running in the online environment of IMS/VS are sharing a single control block structure and thereby avoid the need to reference the VSAM SHAREOPTIONS. The Control Region itself is responsible for ensuring that there will not be any conflicts between the requests made by the various application programs which are accessing the data bases.

The case in which the VSAM SHAREOPTIONS will come into effect is when an attempt is made to share a data base between IMS/VS DB/DC and some other program. This will result in multiple control block structures being used to access the same data set and will cause the SHAREOPTIONS to be invoked.

If sharing of data bases between Online and Batch IMS/VS is desired it will be necessary to define the VSAM clusters with SHAREOPTION 3. This type of sharing is not recommended as it is a part of the philosophy of IMS/VS that the Control Region will have exclusive access to the data bases while the online environment is active. The accepted way in which to allow Batch jobs to process the data bases while they are OPEN to the Control region is to run these batch jobs as BMPs.

It is relatively simple to design the Batch-mode application programs so that they may be run as either normal DB-only batch jobs or as BMPs. This will allow the program to be executed in whichever mode is necessary and will avoid the need to implement user procedures to prevent conflicts between programs.

Potential Exposure Due to Sharing

If it is determined that it is necessary to share data bases between the Online and Batch environments of IMS/VS with a SHAREOPTION of 3, there are a number of areas wherein the data sets may be exposed to error. There are five potential problem areas with this form of sharing.

The first of these is loss of **READ INTEGRITY**. This will occur when there is one application which is updating while another application is reading. Usually, the Online system is performing the updates and the Batch system is processing enquiries against the data base. In this case problems arise when the buffers of the Batch system become unsynchronized with the buffers of the Online system. Loss of synchronization may occur when the reading task fails to reread a control interval and uses an old copy already in its buffers or reads a control interval before the updating task writes the updated version to DASD. This results in enquiries being answered on the basis of out-of-date information and may cause spurious 'record not found' conditions.

The second area of exposure is loss of **WRITE INTEGRITY**. This will happen if both the Batch and the Online systems are updating the data bases at the same time. Performing concurrent update without taking any special precautions to synchronize or serialize the requests can result in ABENDs or permanent corruption of the data set. Updates may be lost through overlapping of requests and multiple concurrent splits in the same control interval may cause corruption of the index. Since VSAM assumes that the information in control blocks is correct and that the content of the buffers is consistent, it is likely that program checks in VSAM could occur.

The third area of exposure is **RBA DISCREPANCIES**. Updates to a data set may cause changes in the High-used, High-key, Sequence Set, or High-level RBA values for the data set[16]. These values are extracted from the catalog records at OPEN time and placed in the control block structure relating to the VSAM data set. Any changes in these values will be made in the control blocks and only be reflected in the catalog once the data set is CLOSEd. This means that programs using the data set will not be aware of any changes that may have occurred due to the activity of another program performing updates. Some of the effects that this may have are incorrect error return codes, incorrect detection of 'add-to-end' processing, incorrect sequential processing and lengthy scanning for direct requests.

---

[16] The meanings of these RBA values are explained in "DATA Component Allocation" on page 126.

The fourth area of exposure to error is **EXTENT DISCREPANCIES**. This type of error may arise if there are multiple concurrent update jobs running against the data base. Since extent information, like the RBA values, is extracted from the catalog, maintained in the control blocks and updated in the catalog at CLOSE, each job will only know about the extents which its own processing has caused. This is likely to result in corruption of the data set because each job will have a different idea about where a particular control interval is located.

5.2 EXCLUSIVE CONTROL CONSIDERATIONS IN VSAM DATA BASES

There are differences in the way in which multiple requests to a data set are synchronized and serialized by Native VSAM and by IMS/VS . A knowledge of these differences is necessary to proper understanding of the processing of VSAM data bases.

5.2.1 Native VSAM Exclusive Control

The facility in VSAM for synchronizing updates across multiple concurrent requests to a data set is known as **Exclusive Control**. This facility is intended to maintain the write integrity of the data set and operates by 'locking' control intervals accessed for update as mentioned above in "Sharing Across Tasks Within a Region" on page 83. Exclusive control is used for concurrent requests to a single control block structure and should not be confused with the SHAREOPTIONS. SHAREOPTIONS only apply when multiple control block structures are OPENed to a single VSAM data set.

Exclusive control is imposed on a control interval whenever a read-for-update request (GET RPL OPTCD=UPD[17]) is issued for a record in that control interval. The control interval will remain in exclusive control status until a subsequent 'write for update' (PUT RPL OPTCD=UPD[17]), 'delete record' (ERASE[17])or 'release control' (ENDREQ[17])request is issued for the affected record. A request to insert a new record (PUT RPL OPTCD=NUP[17]) will also cause exclusive control to be imposed for the duration of the insert operation.

Before VSAM actually places a control interval into exclusive control status, a **look-aside** is performed to determine if any other currently active request already has exclusive control over the desired control interval. If it is found that this is the case, then the request for exclusive control is refused with a logical error which indicates that the record is being held by another request.

---

17 GET, PUT, ERASE and ENDREQ are macro instructions used to make requests to VSAM. OPTCD is a parameter of RPL (Request Parameter List) which indicates, among other things, whether the request is in update (UPD) or nonupdate (NUP) mode.

Read-only requests for data only require shared, as opposed to exclusive, use of the data in the control interval. A read-only request will not result in look-aside to determine if the required control interval is in exclusive control. This is because the exclusive control facility is only concerned with the maintenance of write integrity for the data set and not with read integrity.

When a request is made to a control interval that is already being used by another request, it may or may not be given access to the data depending on whether the request is for exclusive or shared (read-only) control and on the type of control imposed by the request which is currently using the control interval. The rules of access in the case of this sort of conflict is shown below in Figure 36 (see also chapter 'Preventing Deadlock in Exclusive Control' in OS/VS VSAM OPTIONS FOR ADVANCED APPLICATIONS, GC26-3819).

USER B	USER A	
	SHARED request	EXCLUSIVE request
Control Interval is in SHARED control	user A is also given shared access	request is delayed and given ex. access when released by B
Control Interval is in EXCLUSIVE control	request is refused, exclusive control error is returned	request is refused, exclusice control error is returned

Figure 36. Exclusive Control Conflict Resolution: This table shows the result of a request by user A to a control interval which is already in use by user B.

5.2.2 IMS/VS Exclusive Control

The VSAM exclusive control mechanism is only used by IMS/VS when a logical record is to be added to a KSDS (e.g. insertion of a new index pointer segment in a INDEX data base) or an additional control interval must be obtained in an ESDS (e.g. placing a segment into a new block of the Overflow portion of a data base). Other than these cases, the integrity of a data base is maintained by the IMS/VS Program Isolation feature.

5.2.2.1 IMS/VS Integrity Maintenance

IMS/VS allows several requests to access the data in a control interval simultaneously but protects the read and write integrity of the data through a facility known as Program Isolation.

The function of the **Program Isolation Feature** of IMS/VS is to prevent a segment which has been changed from being retrieved or updated by another program until the changed form of the segment has been written to the data base. In order to achieve this effect, IMS/VS uses three levels of control for the segments.

These three levels are:

Read-only This level of control is imposed when a segment is retrieved in nonupdate mode. A segment is released from 'read-only' status when another request to the data base is issued or a synchronization point[18] is reached.

Single-Update This level of control is imposed when a segment is accessed with update intent. A segment is released from 'single update' status when the segment has not been altered and another request to the data base is issued or a synchronization point[18] is reached.

Exclusive If a segment in 'single-update' status is changed by an update or delete request, then exclusive control is imposed on the segment. Exclusive control is released only at a synchronization point[18].

When requests are made to the same segment a certain level of sharing is possible depending on the type of the request and the current status of the referenced segment. If sharing is not allowed because of the interaction of the levels of control, then the request trying to gain access is deferred until control has been released by the previous request. The results of the interaction of the three levels of control is summarized in the following figure.

---

[18] A **synchronization point** is the point at which a program is committed to the correctness of results and the altered control intervals are written out to the data base. A **synchronization point** will be reached when the program terminates, issues a Checkpoint call or, in the case of a program that is processing a transaction that has single processing mode, when a request is made for the next input message.

	segment in READ-ONLY status	segment in SINGLE-UPDATE status	segment in EXCLUSIVE status
request for READ-ONLY control	access allowed the segment is shared	access allowed the segment is shared	access refused the request is deferred
request for SINGLE-UPDATE control	access allowed the segment is shared	access refused the request is deferred	access Refused the request is deferred
request for EXCLUSIVE control	access tefused the request is deferred	access refused the request is deferred	this is an impossible combination

Figure 37. IMS/VS Control Level Conflict Resolution: This table shows the result when a request is made to a segment which is already under the control of another request.

5.2.3 Comparison of Exclusive Control Techniques

The primary difference between Native VSAM and IMS/VS in their techniques of exclusive control is the level at which control is applied:

- With VSAM, the entire control interval is 'locked' which prevents update access to any of the records in the control interval, not merely the record which is being referenced by the 'locking' request.
- IMS/VS, on the other hand, imposes control at segment level and will allow several requests to update data in the same control interval as long as each is working with a different segment.

VSAM has a 'use-count' for each control interval in a buffer in storage to keep track of the number of concurrent requests to that control interval. The buffer containing the control interval will not be written to the data base or reused unless the use-count is zero. This use-count is incremented each time any data in the buffer is accessed at any level of control and is decremented when control is released. In this way, VSAM prevents a control interval from being removed from a buffer while any request is referencing it.

Another important difference between IMS/VS and VSAM exclusive control is that, by its use of the three levels of control and the rules of access shown in Figure 37 on page 93, IMS/VS maintains both read and write integrity while VSAM only maintains write integrity. This may be an important consideration in that, as mentioned earlier in "IMS/VS Exclusive Control" on page 91, IMS/VS will use the Native VSAM exclusive control facility under certain circumstances. One of these cases is the insertion of a new logical record into a KSDS. This will occur when:

- A new root segment is created in an HISAM or a SHISAM data base,
- A new root segment is created in an HIDAM data base resulting in the creation of an index entry in the primary index INDEX data base, or
- A new index source segment is inserted into a data base with a secondary index resulting in the creation of a new index pointer segment in the INDEX data base.

The other case where Native VSAM exclusive control is used is when a new control interval is used in an ESDS. This will occur when:

- A GSAM data base record is extended across a control interval boundary,
- An HISAM data base record is extended into the Overflow area or across a control interval boundary in Overflow,
- A new block in the Overflow portion of an HDAM data base is used to store a segment,
- A new block in a HIDAM data base is used for segment storage, or
- An index pointer segment with a duplicate key is created in an INDEX data which has nonunique keys causing a new control interval in the Overflow portion to be used.

5.3 DATA BASE MANIPULATION WITH VSAM

The purpose of this section is to give the reader a conceptual overview of the activity in the VSAM data sets which takes place when IMS application programs process the data bases. Detailed discussion of internals, such as the mapping of DL/I calls to VSAM requests, is not included.

It is not considered necessary for the user of VSAM data bases to have detailed knowledge of the internal operation of the IMS/VSAM interface in order to be able to understand and use VSAM in the IMS environment.

5.3.1 Insertion

There are a number of different considerations which apply to the insertion of new VSAM records depending on the circumstances under which the insertions are performed. These are treated separately in the following sections.

5.3.1.1 Initial Load Insert

The initial load of a data base is performed by a user supplied program which issues a series of Insert (ISRT) calls to write the data base records segment by segment. This load program must execute as a stand alone batch (DL/I) job and may not be run in the online (DB/DC) environment. Sample load programs may be found in the **IMS/VS INSTALLATION GUIDE, SH20-9081** and in the **IMS/VS PRIMER, SH20-9145**. Depending upon the type of data base being loaded, the VSAM data sets will be processed in different ways.

HISAM and SHISAM

When loading an HISAM or SHISAM data base, the KSDS component is handled in the nonshared resource (NSR) mode and the ESDS component, if present, is handled in local shared resource (LSR) mode. The KSDS is OPENed for keyed sequential access and the ESDS is OPENed for both sequential and direct control interval access.

Since the KSDS is processed with NSR, the **BUFND** and **BUFNI** JCL parameters may be used to allocate buffers for the data set as explained in "VSAM Buffer Use" on page 37.

```
MINIMUM: BUFNI = 1 (default)
         BUFND = 2 (default)
MAXIMUM: BUFNI = 2
         BUFND = 2 * data CI/Track (suggested maximum)
```

Because the data set is being processed with NSR in sequential mode it is possible to make use of VSAM chained write of multiple buffers by supplying additional buffers to improve the performance of the load program. The maximum number of buffers that can be effectively used by VSAM for loading a KSDS is shown above. When this maximum buffer allocation is used, optimal overlap of processing with I/O may occur and VSAM may write a full track at a time.

Since the ESDS is processed by LSR, it is necessary to allocate the buffers by using the **DFSVSAMP** data set as explained in section "VSAM BUFFER POOL Definition Parameters" on page 77. The minimum number of buffers required is 4 and there is no advantage to increasing the number of buffers above this amount. This is because no CI splits can occur in the ESDS and LSR will prevent the VSAM chained write facility from being used.

It does not matter whether the SPEED or RECOVERY options are specified for the ESDS because of the way in which IMS/VS treats the data set. The sequence of operations against the data set is:

1. OPEN for sequential control interval access under NSR.

2. Write a special control record in the first control interval.

3. CLOSE the data set.

4. ReOPEN for direct and sequential control interval access under LSR to insert the actual data base segments.

The second OPEN forces the ESDS to be processed in RECOVERY mode regardless of the option specified in the cluster definition. During the period when the ESDS is processed under NSR when the space map record is being written, the default buffer allocation is sufficient so there is no need to specify a BUFND value for the ESDS.

HDAM and HIDAM

ESDS components of HDAM and HIDAM data bases are processed during initial load in the same way as is the ESDS component of an HISAM data base. The only difference is that the minimum buffer requirement for these data base types is 3 rather than 4.

However, the considerations for the KSDS which serves as the Primary Index of an HIDAM data base are very different from those of the HISAM KSDS. It is not necessary for the user program to explicitly insert the index segments into the Primary Index as they are automatically created by DL/I whenever root segments are inserted into the data base. The Primary Index KSDS is loaded in RECOVERY mode under LSR. This is because of the sequence of operations against the KSDS which are:

1. OPEN for keyed sequential access under NSR.

2. Write a **dummy record** into the data set.

3. CLOSE the data set.

4. ReOPEN for keyed sequential access under NSR.

5. Delete (ERASE) the dummy record previously written.

6. CLOSE the data set again.

7. ReOPEN for direct and sequential control interval access under LSR to load the actual index segments.

The ReOPENing of the KSDS forces it into RECOVERY mode regardless of the option specified in the cluster definition.

Since the data set is being ultimately loaded in the LSR environment, its buffers must also be allocated by subpool definition in the DFSVSAMP data set.

The number of buffers required for the KSDS is 3. It is of no benefit to assign more than 3 buffers as no CI splits can occur during initial load and LSR prevents the use of VSAM chained writes.

For the period when the KSDS is being processed under NSR, the default buffer allocation is sufficient so there is no need for the user to be concerned with specification of BUFNI or BUFND.

5.3.1.2 Direct Processing

There are two forms of direct processing available with VSAM, **keyed** and **addressed**. With keyed direct processing records can be retrieved, updated, deleted and added. A key value must be presented by the user for each logical record that is to be processed. For retrieval operation, the supplied key can be the full key, or a **generic key** (the leftmost part of the key) which matches exactly or is less than the key of the desired record, and the record retrieved will have the exact or next greater key. When inserting a new record, the full key must be presented as part of the record to be inserted. The primary index is searched from the top to the bottom level to locate the requested logical record or the point at which to insert the new record.

Keyed direct processing will be used when accessing root segments in HISAM or SHISAM data bases. It will also be used when accessing index segments in an INDEX data base in order to process an HIDAM root or a target segment in a data base which has a secondary index.

With addressed direct processing records may be retrieved, updated, deleted or added in a KSDS and retrieved or updated in an ESDS. The user provides the Relative Byte Address (RBA) of the logical record or control interval that is to be processed rather than a key. There is no index search with addressed direct processing as the specified RBA is translated directly into a physical DASD address.

Addressed direct processing is used for the ESDS components of data bases and is the basic mechanism for accessing segments in Hierarchical Direct organizations where the pointers are RBAs.

5.3.1.3 Sequential Processing

For sequential processing the keyed and addressed forms are also available. The basic difference between the two forms is that logical records are accessed in key sequence with the keyed form and in physical storage sequence with the addressed form.

Keyed sequential processing is used to load a KSDS and to retrieve, update, delete and add logical records to an existing KSDS. When keyed sequential processing is used, records can be processed in ascending or in descending sequence by primary key. When retrieving records, key values need not be user-supplied, since VSAM automatically obtains the next logical record in sequence.

Following types of operations can be performed using key-sequential processing:

- Records can be processed in ascending primary key sequence. This is called **forward processing.**
- Records can be processed in descending primary key sequence. This is called **previous record processing** or **backward processing.**
- A **mass sequential insertion** technique is used by VSAM when additions of contiguous records are sequenced and sequential processing is used.

For key-sequential processing the index sequence set is used to find the next logical control interval.

Keyed sequential forward processing is used for initial load of the KSDS portions of data bases and when processing Get-Next calls. It is also used for insertions when INSERT=SEQ has been specified on the **OPTIONS** statement as explained in "OPTIONS Statement" on page 80. Backward processing is only used by IMS for Recovery or Restart, when the SIP flag has been found on, in order to complete an interrupted CI split.

Addressed sequential processing is only used when a backup copy (Image Copy) of a data base is created. The ESDS components of data bases are loaded and accessed using direct addressed processing.

Search arguments are not required or used with either form of sequential processing.

5.3.1.4 Skip-sequential Insertion

Keyed skip-sequential processing, is a variation of direct processing. It can be used for retrieval, update, addition and deletion operations. The logical records must be processed in ascending sequence without pre-positioning. Only the sequence set of the primary index is used for skip-sequential processing based on the primary key.

When a relatively small number of transactions that are in key sequence and clustered in sequence are to be processed, skip-sequential processing can be used to retrieve the records directly by key.

Skip sequential processing can be used to avoid retrieving the entire data set sequentially to process a relatively small percentage of the total number of records, or to avoid using direct retrieval of the desired records, which causes the primary index to be searched from the top to the bottom level for each record retrieved. However skip-sequential does read all sequence set records between those in question, therefore it is not suggested that skip-sequential be used when records from the beginning and from the end of the data set are to be processed.

Skip-sequential processing will be used for insertion of HISAM and INDEX root segments when INSERT=SKP is specified on the OPTIONS statement. The OPTIONS statement was discussed earlier in section "OPTIONS Statement" on page 80.

5.3.1.5 Secondary Index Segment Insertion

A secondary index segment will be automatically created by DL/I whenever a new index source segment is introduced into the data base being indexed. Regardless of the mode in which the source segment is inserted, the actual secondary index segment will be created using keyed direct access if the index has unique keys or it is the first occurrence of a nonunique key.

This form of insertion is used because these segments reside in the KSDS portion of the INDEX data base and must be maintained in key sequence. If the index segment has the same key as an existing index segment, then it will be inserted into the ESDS portion of the data base using addressed direct processing.

5.3.2 Update

The basic unit of information in an IMS data base is the segment and all DL/I requests are issued in terms of segments. For VSAM, the basic unit is the logical record and all VSAM operations are performed in terms of logical records. Since each VSAM logical record may contain a number of data base segments and associated control information, updating of VSAM logical records will take place under a number of different circumstances in an IMS/VS environment.

VSAM records will be updated when:

1. A segment's contents are changed by a REPL (Replace) call. If the size of a variable length segment is changed by REPL, then multiple updates may occur.

   In an HISAM data base segments are stored in hierarchical sequence so that increasing a segment's length may cause subsequent segments to be 'spilled' to the next VSAM record which forms part of the data base record. Both records must then be updated to show the new version of the replaced segment **and** the movement of the 'spilled' segment.

   For HDAM and HIDAM data bases, segments are not moved unless the increase in size of a segment is such that it can no longer fit into the record which it previously occupied. In this case, the segment must be relocated to some other record and both of these records are updated.

2. A dependent (non-root) segment is inserted into a data base and is placed in a logical record which already contains a segment.

   This can also result in multiple updates taking place. It may cause segments in an HISAM data base to be shifted between records. In HDAM or HIDAM data bases the pointers in the prefixes of segments located in other logical records may have to be updated to preserve the relationships between segments.

3. A segment is deleted from a data base.

   Segments are often only deleted in the sense that they can no longer be retrieved by a GET call but are **not** physically removed from the data base. This form of deletion is accomplished by updating the segment to turn on the appropriate flags in the Delete Byte which is the first part of the segment prefix.

   When a segment is to be physically removed from the data base, the VSAM logical record which contains the segment is updated and the space which the segment had occupied is made available for use by other segments.

Updating will be performed in the appropriate mode as described previously in sections "Direct Processing" on page 97 and "Sequential Processing" on page 98.

5.3.2.1 Replace of Secondary Index Source Segment

If the fields in a secondary index source segment which form the key of the secondary index are changed as the result of a REPL call, there are special considerations which apply to the corresponding segment in the INDEX data base.

The INDEX data base segment is **updated** during **ONLINE** processing only.

During **BATCH** processing the INDEX data base segment is **erased** (physically removed from the data base) by keyed direct processing using the old key and then a new segment is inserted with the new key. This is done in order to insure that the segments in the INDEX data base are maintained in the proper sequence by key.

The only cases in which a segment in an INDEX data base will be updated are when the duplicate data fields[19] are changed or the INDEX data base is processed directly by an application program.

5.3.3 Deletion

As mentioned above in section "Update" on page 100, in most cases segment deletion consists of setting on flags in the Delete Byte with a VSAM update request. Even when a segment is deleted and physically removed from the data base, it will usually be done by updating the logical record which contains the segment. There are, however, certain cases in which DL/I deletion will use the VSAM **ERASE** function.

VSAM ERASE causes a KSDS logical record to be eliminated from DASD storage and the space it occupied made available for reuse as VSAM free space. ESDS records may not be ERASEd.

If processed in a DL/I **BATCH** environment DL/I will issue an ERASE request for records in the KSDS component of a HISAM or INDEX data base.

In an **ONLINE** environment, deletion would result in updating the logical record (no ERASE).

A SHISAM data base constitutes a special case where ERASE is not only done in a BATCH environment but **also** in an ONLINE environment if the processing option of **DELETE** (PROCOPT=D) is specified. In this case, scheduling will only allow concurrent Read-Only access.

---

[19] Duplicate data fields are fields from the secondary index source segment which the user has requested to be stored in the index segment. These fields are not used for retrieval purposes and are only available to the user when the INDEX data base is processed directly.

5.3.3.1 Crawling

There is another problem with key design that may lead to a situation where the data set is effectively **crawling** across the disk. Crawling occurs when records are inserted with steadily increasing key values and only low key records are deleted.

When all of the records in a CI have been deleted, VSAM does not return the CI to the free status nor does it delete the index entry for the CI from the sequence set record. This means that the space in the CI can only be reused for records whose keys are in the same range as those originally located in the CI.

If only high key records are inserted and only low key records are deleted, then the space released by deleted records will never be reused because no records with similar keys are being placed in the data set. The data set will continue to extend at its high order end in order to accommodate the new high key records. To the user, this data set will appear to be continually growing even though the number of records in it remains fairly constant. It is actually crawling forward across the disk, taking up space for new records at one end and leaving unuseable space behind.

There are two basic characteristics of a 'crawling' data set. The first is that records are not permanently kept in the data set but are maintained for a certain period of time before being deleted. The second is that the records are inserted with ever increasing key values. It is quite common to find applications where records have a period of currency and are then discarded but crawling can be prevented in these cases by avoiding the use of ascending keys. Continually increasing key values are most commonly caused by the high order portion of the key being a date or timestamp field.

Note: Data set space can be reclaimed by reorganizing the data set (e.g. with REPRO).

6.1 GENERAL DESCRIPTION OF FAST PATH FEATURE

The IMS/VS Fast Path feature offers a means of incorporating high volume limited function transactions in a single DB/DC system. The user can select the Fast Path feature for those applications that need simple data structures and where there is a requirement for high volume of transactions.

Fast Path (FP) achieves high transaction rates by exchanging functions for speed of processing. This results in the use of new facilities that are briefly described below.

- Two new types of data base:

  Main Storage Data Base (MSDB) - a root-only data base for keeping heavily used data in main storage thereby, reducing input/output activity.

  Data Entry Data Base (DEDB) - a new type of data base that provides root segment and up to seven dependent segment types and uses VSAM Improved Control Interval Processing.

  In addition new techniques are used to reduce contention between concurrently scheduled transactions.

- Expedited Message Handler (EMH):

  Fast Path uses incore message queues exclusively, the IMS/VS Message Queue is not involved. Multiple copies of the same application program operating in wait-for-input mode can be associated with each incore message queue. The design of Fast Path also allows more parallel processing of tasks, especially on a multiprocessor system.

- Mixed Mode:

  This term refers to the fact that FP and IMS/VS functions are fully integrated within one single system. This means that FP transactions have access to DL/I data bases and to current IMS/VS facilities; conversely, IMS/VS transactions have access to FP data bases and facilities.

6.2 DEDB

The DEDB characteristics and its performance capabilities are best introduced by first comparing VSAM Normal Control Interval processing with VSAM Improved Control Interval processing.

6.2.1 Improved Control Interval Processing (ICIP)

VSAM ICIP has fewer options and is more restrictive than the Normal Control Interval processing but this is also the reason why VSAM ICIP requests have shorter path lengths. The following table lists the main differences between the two processings.

Normal Control Interval Processing	Improved Control Interval Processing
CI sizes: n*512 bytes if < 8192, then n*2048 bytes up to a max. of 32,768 bytes	CI sizes: 512, 1024, 2048, and 4096 bytes
VSAM or user buffering	user buffering only
asynchronous or synchronous requests	synchronous requests only
direct or sequential requests	direct requests only
unfixed control blocks and buffers	optional fixed control blocks and buffers
chained-RPL's	single RPL only
non-authorized user	authorized or non-authorized user
update in place or add-to-end	update in place only
SMF EXCP counts	no SMF EXCP counts
TCB processing	TCB or SRB processing

Figure 38. Fast Path ICI Processing Comparison

From the above comparison, one can deduct immediately that the Data Base Administrator will have fewer choices of CI sizes (512, 1024, 2048, and 4096 bytes), and because VSAM ICIP supports update in place only, the data base data sets will have to be preformatted.

VSAM control blocks can be optionally page fixed by requesting that DEDB control blocks be page fixed by IMS.

Other restrictions have been taken care of during the design and implementation of the Fast Path product and do not concern the user directly.

6.3 DEDB CONCEPT

A DEDB is an HD-type data base containing up to eight segment types in a two level data base structure (Figure 39). The possible combinations of segment types are:

- A root segment with a sequential dependent segment with up to six direct dependent segment types.
- A root segment with maximum of seven direct dependent segment types.

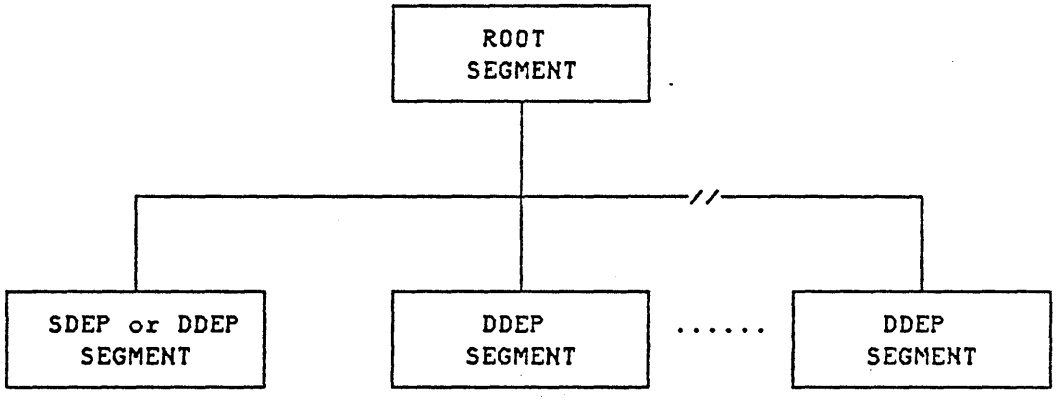


Figure 39. DEDB Record Structure

Root segments and **direct dependent** (DDEP) segments are similar to DL/I data base segments. Retrieve and update calls are a compatible subset of the current DL/I calls. As in HDAM, root segments are chained off a Root Anchor Point (RAP) and a randomizing routine is used to derive a RAP number.

Sequential dependent (SDEP) segments are of a new type that is especially designed to provide fast insert capability. Their intended use is to collect data very quickly and insert them in mass in the data base cutting down the number of I/O operations that would normally be required with other data base organizations. Sequential dependent segments can only be inserted and retrieved, no update capability is available.

Variable length segments are fully supported. Because VSAM ICIP uses User Buffering only, VSAM is not concerned with variable length data. Each CI consists of one logical VSAM record, one RDF, and one CIDF.

FP does the management within the CI buffers and merely request VSAM to read or write those buffers using ICI processing.

There are several features included in DEDB for optimization processing and extended operation. Described here are those that pertain directly to the use of VSAM ICIP by the Fast Path feature.

6.3.1 Fast Path use of VSAM ICIP

The MVS implementation of the Fast Path feature is based on SRB processing which has a significantly shorter path length than TCB processing. TCB processing of the DEDB is used in the OS/VS1 implementation.

To compensate for the fact that ICIP requests have to be synchronous, the implementation allows multiple parallel requests to take place concurrently.

Read access to the DEDB is done directly from the dependent regions. Therefore multiple requests can be run in parallel. The unit of allocation is the CI. Thus if there is no contention at the CI level, the level of parallelism that could potentially be achieved is equal to the number of dependent regions active in the system.

On the output side, the implementation is different. CIs are not written back in the data base during the life of the transaction but after its successful completion. This has two advantages: first, in case of an ABEND, no backout of the data base is necessary, second, region occupancy is lower.

To free up the dependent region as soon as possible, CIs are transferred to an SRB (MVS) processing called Output Thread that invokes VSAM ICIP.

Multiple Output Thread can be defined up to a maximum of 10 to allow for more parallel processing. Each Output Thread handles all the CIs updated by a single transaction. Because of VSAM ICIP, each CI is written by one request at a time, the DEDB action module is then internally redriven until all the CIs on the Output Thread have been written back. At which time, the Output Thread is made available for further processing.

Because of VSAM ICIP, AREAs cannot be extended. Out-of-space conditions can happen in the root addressable part and in the sequential dependent segment part. Appropriate warnings are sent to the master terminal operator when those conditions arise.

A **/DISPLAY** command and a new data base call (POS) can be used at any time to evaluate the utilization of both parts.

The following considerations are mainly concerned with the VSAM aspect of the DEDB data base definition. Discussed are the sizes of CI and UOW, followed by comments on the AREA concept.

6.3.1.1 DEDB CI Size

The choice of a CI size depends on the following factors:

- Because of VSAM ICIP, four different CI sizes are supported. A choice must be made between 512, 1024, 2048, and 4096 bytes CI sizes.

- There is only one RAP per CI. The average record length has to be taken into account. In the base section of the root addressable part, a CI can be shared only by the roots, which randomize to its RAP and their DDEP segments.

- CSA main storage availability. It should be noted that the largest CI size among all the opened DEDBs sets the size of the FP buffer. There is only one FP buffer pool used for processing all DEDB data bases. Although CI sizes can vary between AREA(s) and DEDBs, a common CI size will save storage space.

  If only one AREA is defined with a CI size of 4096 bytes, no matter what the other CI sizes are, the entire buffer pool will be set up with 4096 bytes buffers.

- Track utilization according to the device type.

- Performance of sequential dependent segment writes. A larger CI requires a fewer number of I/Os to write the same amount of sequential dependent segments.

- The maximum segment size which is 3976 bytes if using a 4096 CI size.

6.3.1.2 DEDB Unit-of-Work (UOW) Size

The UOW is the unit of space allocation by which one specifies how large the root addressable and the independent overflow parts are to be.

Three factors might affect the size of the UOW:

- The DEDB Direct Reorganization Utility runs on a UOW basis. Therefore, while the UOW is being reorganized, none of the CIs and data they contain are available to other processing.

  A large UOW could cause resource contention, resulting in increased response time if the utility is run during the online period.

  A small UOW could cause some overhead during reorganization as the utility switches from one UOW to the next one with very little useful work each time. But this might not matter so much if reorganization time is not critical.

- Another factor that could affect the size of the UOW is the sequential processing of DEDB using FP non-message-driven regions or BMP regions. There is a facility known as the processing option 'P' that can be used to speed up sequential processing of DEDB in batch mode. This facility is directly related to the size of the UOW. Refer to the **FAST PATH FEATURE DESCRIPTION AND DESIGN GUIDE, G320-5775** for a discussion on this subject.

- The dependent overflow (DASD space) usage is more efficient with a large UOW than with a small UOW.

6.4 AREA CONCEPT

In DL/I data base organizations, a large data base implies a large data set, or, if multiple data sets are used, the data record structure is broken up on a segment basis and pieces of the same data base record are found in these multiple data sets.

DEDB implements multiple data sets called AREAs. But the major difference with DL/I data base organization is that each data base record is entirely contained within one of the data sets as shown by Figure 40. Each AREA is a VSAM ESDS that is managed by FP using VSAM ICIP. This concept leads to some very valuable advantages that are described below especially in the area of extended operation with large data bases.

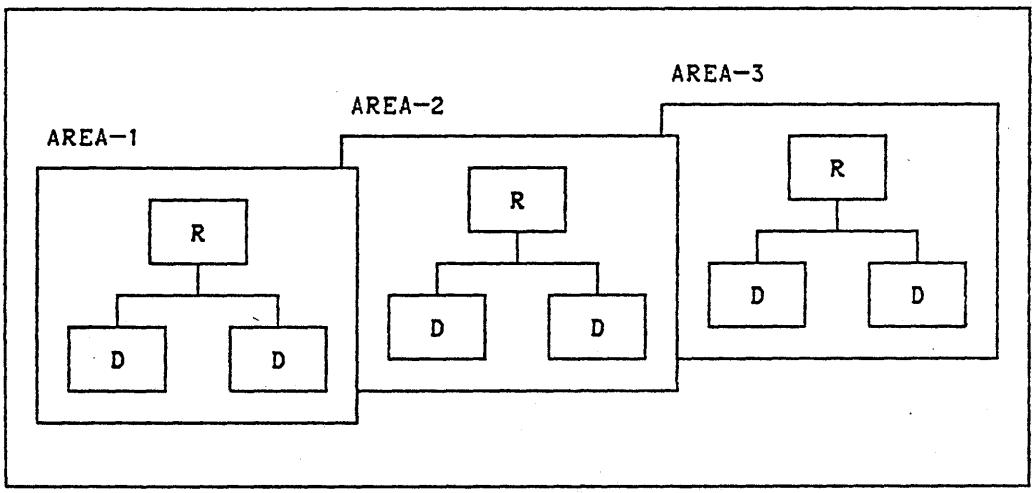


Figure 40. AREA Concept

Space management parameters can vary from one AREA to another. This includes: CI size, UOW size, root addressable part, overflow part, and sequential dependent part. Also, the device type can vary from one AREA to the other.

It is feasible to define an AREA on more than one volume and have one of them dedicated to the sequential dependent part. This implementation might save some seek time as sequential dependent segments are continuously added at the end of the sequential dependent part. The savings will depend on the current size of the sequential dependent part and on the blocking factor used for the sequential dependent segments. If an AREA spans more than one volume, these volumes must be of the same type.

VSAM control blocks can optionally be page fixed by requesting from IMS/VS that DEDB control blocks be page fixed. This is done by specifying the DEDB parameter in the procedure member DFSFIXnn. As part of the open processing of an AREA, the system will then modify the corresponding ACB macro to include the CFX option. This in turn will request from VSAM that control blocks be page fixed.

VSAM ICIP requires AREAs to be preformatted. This means that a VSAM Define must be followed by the execution of the DEDB Initialization Utility. Each invocation of the utility allows all AREAs of one single DEDB to be initialized.

The designer should keep in mind that the DEDB is not extendible. Sufficient space must be provided for all parts. A /DISPLAY command and a new data base call (POS) can help monitor the usage of auxiliary space. Unused space in the root addressable and in the independent overflow parts can be reclaimed through reorganization using the DEDB Direct Reorganization Utility.

6.4.1.1 System

The user must specify at IMS/VS initialization time the characteristics of the buffer pool to be used in part to read and write DEDB CIs. The buffer length specified must be large enough to accommodate CIs of any AREAs to be opened while the system is active. VSAM ICIP assumes buffers are of the correct size, a check will be made by IMS/VS at OPEN time for this purpose and the OPEN will be rejected if the buffer size specified is too small.

6.4.2 Space Allocation in Units-of-Work

The DEDB AREA is divided into three parts:

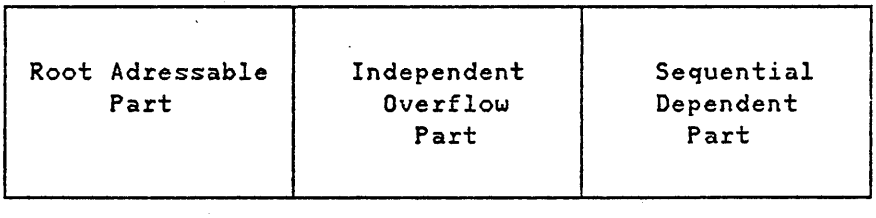


Figure 41. AREA Description

6.4.2.1 Root Addressable Part

The root addressable part itself is divided into Units-of-Work (UOW) which are the basic elements of space allocation and also the basic unit of reorganization.

A UOW consists of a user-specified number of VSAM control intervals located physically contiguous. The root addressable part contains root segments and direct dependent segments.

Each UOW in the root addressable part is further divided into a base section and an overflow section.

The base section contains those CIs of a UOW which are addressed by the randomizing module, whereas the overflow section of the UOW is used as a logical extension of a CI within that UOW (see Figure 42).

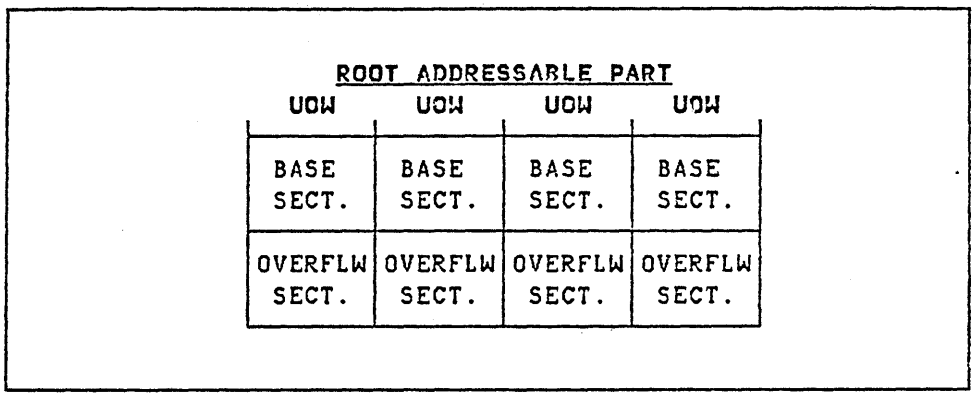


Figure 42. UOW Concept

6.4.2.2 Independent Overflow Part

Root and direct dependent segments that cannot fit in the root addressable part are written in the independent overflow part.

Any empty control interval (CI) in the independent overflow part can be used by any UOW. However, once a CI has been obtained by a UOW, only segments belonging to this UOW can be stored therein.

6.4.2.3 Sequential Dependent Part

Sequential dependent segments are entered in this part in a time-ordered sequence without regard to the UOW containing the root segment. When this space is filled, it is reused in a wrap around manner. Before this happens, sequential dependent segments can be mass retrieved using the DEDB Scan Utility. Then the DEDB Delete Utility should be invoked to delete a logical contiguous portion or all of the sequential dependent segments.

6.4.3 DBD Macro

The DBD description is essentially the same as for DL/I data bases with the exception of the AREA statement. An AREA statement is required for each AREA of a DEDB.

```
AREA  DD1=ddname,DEVICE=device,MODEL=model,SIZE=size,
      UOW=(number1,overflow1),ROOT=(number2,overflow2)
```

Explanation:

DD1=ddname 'ddname' references the JCL DD statement defining the VSAM ESDS which corresponds to this AREA.

SIZE=size The 'size' of CI in bytes size can only be: 512, 1024, 2048, or 4096 bytes. No default value is allowed. 2048 cannot be specified with 3340 and 3344.

UOW=(number1,overflow1) 'number1' specifies the number of VSAM CIs in a UOW. Its value must be in the range from 2 to 32,767.

'overflow1' represents the number of CIs in the overflow section of a UOW.

'overflow1' can be any value greater than or equal to one but at least one less than 'number1'.

ROOT=(number2,overflow2) 'number2' specifies the total number of UOWs to be allocated to the root addressable and independent overflow parts. The rest of the VSAM data set is reserved for the sequential dependent segment part. The value must be at least '2', and cannot be larger than the amount of space actually in the VSAM data set.

'overflow2' specifies the number of UOWs reserved for the independent overflow part. It must be at least '1' and must be less than the value specified for number2.

6.5 ADVANTAGES OF DEDB

The AREA concept, as explained above, leads to the following advantages:

- The division of the data base into AREAs is transparent to the application program.
- Large data bases can be implemented (many times bigger than the 2**32 byte limitation for a single VSAM data set.)

- A maximum of 240 AREAs per DEDB is allowed.

- AREAs have a useful degree of independence. Not all AREAs need be online and the master terminal operator has the facility to start or stop an AREA. In an MVS environment, dynamic allocation/deallocation at the DEDB/AREA level is supported.

- The impact of permanent I/O errors and catastrophic failures is reduced with DEDB. DL/I requires that all data base data sets be available all the time. With DEDB the data not available is limited only to the AREA affected by the failure. Because all DEDB utilities run at the level of the AREA, the recovery of the failing AREA can be done while the rest of the data base is accessible to online processing. The currently allocated log volume will have to be freed by a /DBR AREA command and used in the recovery operation. Track recovery is also supported. The recovered AREA can then be dynamically allocated (MVS) back to the operational environment.

- Each AREA can have its own space management parameters. The user may choose these parameters according to the message volume which may vary from AREA to AREA.

- AREAs of a DEDB can be allocated on different volume types.

- All maintenance and recovery operations are on an AREA basis, instead of on a whole data base.

- Maintenance operations like reorganization, retrieve, and delete of sequential dependent segments are all done online, increasing the availability of the data to the network.

6.6 RESTRICTIONS WITH DEDB

The following is a list of some limitations or restrictions pertaining to DEDB compared with other DL/I data base organizations:

- As shown before, the data base record structure is very simple and the number of segment types are limited.

- There is no update capability against the sequential dependent segment type but this is not necessarily a limitation, it could be regarded as an additional security facility.

- DEDB data bases can only be processed online, there is no offline processing capability.

- The maximum segment size allowed is 3976 bytes if using a 4096 bytes CI size.

7.1 CAPABILITIES OF ACCESS METHOD SERVICES

Access Method Services is the basic VSAM utility program used to create and maintain all VSAM data sets. It serves as a single interface for the user to manipulate catalog information and the structure and content of data sets.

A large number of different services are provided by Access Method Services, each of which is invoked by a special command. This avoids the need for the user to contend with multiple utility programs and their individual formats and requirements as is the case with the OS Utilities.

The various commands available under Access Method Services are described below.

7.1.1 DEFINE command

This command is used to create catalogs, and catalog entries for clusters, data spaces, and nonVSAM data sets. In MVS, catalog entries for ALIASes, GDGs, and PAGESPACEs can also be created. DEFINE creates the catalog entry for a VSAM object and allocates space for this object.

All VSAM data sets **must be DEFINEd** before they can be loaded or used in any way.

For correct coding see **OS/VS1 ACCESS METHOD SERVICES, GC26-3840** or **OS/VS2 ACCESS METHOD SERVICES, GC26-3841.**

7.1.2 REPRO

This command copies records from one data set to another. All or a selected portion of the records in a data set may be copied in key sequence or in physical sequence.

The records to be copied may be identified by keys (FROMKEY, TOKEY), by Relative Byte Addresses (FROMADDR, TOADDR), or by record count (SKIP, COUNT).

If the user does not specify that a particular portion of a data set is to be copied, then the entire data set is copied.

When records are transferred into an empty data set, **REPRO** performs an initial load and creates a reorganized copy of the original data set.

The different functions of REPRO are:

- Add records to the end of an ESDS
- Copy a VSAM catalog (MVS only) (move a catalog to another disk)
- Load/copy data sets

```
ISAM ——→ SAM    (backup/unload an ISAM data set)
ISAM ——→ VSAM   (convert an ISAM data set to VSAM format)
SAM  ——→ SAM    (copy, e.g. tapes)
SAM  ——→ VSAM   (load a VSAM data set)
VSAM ——→ SAM    (backup/unload a VSAM data set
VSAM ——→ VSAM   (copy/merge data)
```

- Merge records into KSDS or RRDS
- Punch or print

```
ISAM data sets
SAM  data sets
VSAM data sets
```

- Unload/reload a VSAM catalog (backup)

7.1.3 PRINT

This command is used to print ISAM, SAM or VSAM data sets, or parts of these data sets. Selection of the part of the data set to be printed may be made in the same way as is done for REPRO. The printout can be obtained in hexadecimal, character or dump format.

7.1.4 EXPORT/IMPORT

These commands are used to unload/reload VSAM data sets for backup and/or transportation.

The EXPORT command has, among others, two important options: **TEMPORARY|PERMANENT**. EXPORT **PERMANENT** unloads the data set and deletes the entries from the catalog. EXPORT **TEMPORARY** produces an unloaded copy of the data set, and marks the catalog entry, to show that a temporary copy exists. In the unloaded format, the data set is not accessible.

EXPORT uses the VSAM catalog to copy all related information such as cluster entry, data entry, and index entry (for KSDS only) from the catalog to the target data set, a SAM VBS data set (allocated by the user via JCL).

IMPORT redefines and reloads the data set. Redefinition is done by restoring the entries copied out of the catalog by EXPORT. If the entries still exist in the catalog, IMPORT will replace them with the EXPORTed versions. IMPORT will also produce a reorganized form of the data set by reloading all of the records in the proper sequence. IMPORT can be considered equivalent to a REPRO preceeded by an internal DEFINE.

These commands can also be used to disconnect/reconnect a user catalog from/to a VSAM master catalog (if a user catalog is disconnected from the master catalog by deleting its connector entry, it cannot be accessed from that system). No unload/reload is necessary.

7.1.5 LISTCAT

This command is used to list the VSAM catalog entries or parts of them. It is recommended that a LISTCAT command be executed after any DEFINE command in order to produce a hardcopy record of the options selected by the user or chosen or overridden by VSAM. Examples and explanations of LISTCAT output are included in section "VSAM Catalog Information" on page 125.

7.1.6 VERIFY

This command is used to ensure a catalog reflects the correct 'High-Used RBA' of a data set (the 'High-Used RBA' points to the last byte used in the VSAM data set; see "DATA Component Allocation" on page 126). VERIFY should be used after an OPEN-error caused by a previous system failure or an ABEND condition while updating the data set.

VERIFY **cannot** be used for empty KSDS data sets (identified by High-Used RBA = 0 in the catalog cluster entry). This condition will occur when an ABEND or system failure occurs during the load of the data set with the SPEED **or** RECOVERY option specified (see section "SPEED/RECOVERY" on page 70), or while reloading a reusable data set specifying the REUSE option. VERIFY and its usage in IMS/VS are also discussed in "VERIFY" on page 122.

7.2 ACCESS METHOD SERVICES FUNCTIONS RELATED TO DATA BASES

Access Method Services is a valuable addition to the inventory of tools available to the IMS/VS data base user. It has certain unique capabilities that are not available through the normal set of IMS Utilities which are particularly useful when dealing with data bases consisting of VSAM data sets. In addition, there are circumstances in which it may be advantageous to use Access Method Services **instead of** the IMS Utilities to perform the same functions. In the following sections, those facilities of Access Method Services which relate specifically to the data base environment are discussed.

7.2.1 Backup

One of the most important considerations in maintaining the integrity of a data base is the provision of an adequate level of backup. Backing up a data base essentially consists of making a complete and accurate copy of all of the information stored in the data base.

The IMS/VS system supplies a special purpose program for making back up copies of data bases, the **IMAGE COPY Utility**. This utility is designed to interface with the Restore and Recovery facilities of IMS/VS in order to produce a complete data base management service. However, IMAGE COPY is not the only acceptable way to back up a data base. Access Method Services' REPRO and EXPORT commands may also be used for this purpose.

7.2.1.1 Backup by REPRO

REPRO may be used both to create a backup copy of a data base and to subsequently restore the data base from the backup copy. Before the backup may be restored, the data set must be emptied by either deleting and redefining the data set or by specifying the REUSE option. If no records have been either added to or deleted from the data set between the time that the backup was taken and the time that it is restored, then the REPLACE option may be used. For further information on the REUSE and REPLACE options of REPRO, refer to the appropriate Access Method Services manual.

The ability of REPRO to selectively copy portions of a data set is not greatly useful for backing up data bases. Although it is possible to conceive of cases where a partial backup may be desirable, the usual requirement is for a copy of an entire data base.

In order to produce a backup copy of a data base **all** of the data sets which comprise the data base must be individually REPROed. It is important that no updating be allowed against the data base until all of the data sets have been copied, otherwise there may be a loss of synchronization between the various portions of the data base. This effect can be achieved within a single CPU by using SHAREOPTION 1 as explained in "VSAM Sharing" on page 83.

The time required to take a backup with REPRO may be reduced by running the REPROs for the data sets in parallel and by allocating sufficient buffers with the **BUFNI** and **BUFND** JCL parameters. The recommended number of buffers for REPRO is:

```
BUFNI = 2
BUFND = 2 * data CIs/track
```

There are two drawbacks to the use of REPRO for backup purposes. The first is that it only copies the records from a data set with the result that, when the backup is restored, statistical and status information from the VSAM catalog is lost. The second is that, when the data set is restored with REPRO, it is reorganized. Reorganization will change the RBA values for KSDS records which may cause difficulties when Recovery or Backout processing is attempted. ESDS records are never relocated by reorganization so there is no danger of RBA discrepancies for this type of data set. These difficulties arise because Recovery and Backout are based on the log tape where many of the data base changes are recorded in terms of RBAs rather than by keys.

7.2.1.2 Backup by EXPORT

Since EXPORT creates a copy of all records in the data set and also copies the catalog information for the data set, it is more suitable for Backup than is REPRO. This is because copying the catalog entry along with the data set will ensure that the two remain in synchronization thereby avoiding one of the problems which may occur with REPRO.

Any data set which has been backed up with EXPORT **must** be recovered by IMPORT. Unlike REPRO, there is no need to empty the data set before IMPORTing the backup copy since IMPORT will perform a complete replacement of the data set and its catalog information in every case. However, this does mean that the data set will be reorganized after IMPORT and may result in Recovery and Backout problems.

Each data set in a data base must be individually EXPORTed for backup and IMPORTed for restore. No updating of the data base should be allowed during the period of EXPORT or of IMPORT. SHAREOPTION 1 is also applicable to this case and will ensure that no such updating takes place.

The performance of EXPORT and IMPORT may also be improved through parallel scheduling and buffer allocation. The buffer recommendations for these two commands are exactly the same as for REPRO.

7.2.1.3 Comparison With IMAGE COPY Utility

There are two forms of the IMAGE COPY Utility, Batch and Online. Both forms of IMAGE COPY have the ability to create 1 or 2 backup copies of the data sets in a data base as opposed to REPRO and EXPORT which can only create 1 copy at a time. The major differences between the Batch and Online versions of Image Copy are:

1. Batch runs as a Stand Alone (DL/I) program while Online runs as a BMP under the IMS/VS Control Region.

2. Batch buffer allocation is performed via JCL parameters while Online buffers must be allocated in the DFSVSMxx member of the IMS procedure library as explained in "VSAM BUFFER POOL Definition Parameters" on page 77.

3. A DD statement for each data set to be copied must be included in the JCL for the Batch version but this is not necessary for the Online version. This is because Online Image Copy interfaces with the Control Region where the data sets are already allocated.

4. Batch can produce copies of all of the data sets in a data base in a single invocation while Online copies each data set separately.

Both forms of IMAGE COPY create copies of the data sets by writing them out, control interval by control interval, in physical sequence. They also include certain special control records in the output copy of the data set for use during restoration. When Image Copies are restored, the data sets are **not** reorganized but are recreated in their original form. This means that Recovery and Backout can be applied directly to a restored IMAGE COPY without fear of problems due to RBA changes. However, Image Copies do not reset the VSAM catalog information when restored so there is a chance that the catalog entries may be incorrect after restore.

For further information on the IMS IMAGE COPY facility, refer to the **IMS/VS UTILITIES REFERENCE MANUAL, SH20-9029.**

Each of the three ways of taking Backups has advantages and disadvantages which must be considered when choosing a Backup technique. REPRO is potentially the fastest of the three since it has the least overhead. On the other hand, REPRO also has the greatest exposure to problems with Recovery and Backout processing and loss of catalog information. EXPORT and Image Copy perform more functions than does REPRO and may therefore execute more slowly. However, EXPORT preserves catalog information and IMAGE COPY is suitable for subsequent Recovery and Backout processing.

If the necessary time and resources are available, Backup should be done by **both** EXPORT and IMAGE COPY for maximum security. The EXPORT copy can then be used when the data base needs to be reset but does not require Recovery or Backout while the IMAGE COPY will be used when Recovery and Backout are necessary. Furthermore, by taking Backup with two different methods and keeping two copies of the data bases, the chance of some sort of failure which prevents restoring a data base is greatly reduced.

7.2.2 Reorganization

When a data base experiences a great deal of insert and delete activity, data base records may become badly fragmented and space may be lost due to its occupation by deleted segments. This will cause a loss of processing efficiency and may severely impact response times. In order to reclaim lost space and bring the data base records back into hierarchical sequence, the data base must be **reorganized.**

7.2.2.1 Function of IMS/VS Re-org Utilities

The IMS/VS system provides ten utility programs which are used for data base reorganization:

1. **HISAM Reorganization Unload (DFSURUL0):** used to unload HISAM, SHISAM and HIDAM Primary Index data bases to a QSAM formatted data set.

2. **HISAM Reorganization Reload (DFSURRL0):** used to reload HISAM, SHISAM and HIDAM Primary Index data base from the QSAM formatted data set created by HISAM Reorganization Unload.

3. **HD Reorganization Unload (DFSURGU0):** used to unload HDAM, HIDAM or HISAM data bases to a QSAM formatted data set.

4. **HD Reorganization Reload (DFSURGL0):** used to reload HDAM, HIDAM or HISAM data bases from the QSAM formatted data set created by HD Reorganization Unload.

5. **Data Base Surveyor Utility Feature (DFSPRSUR):** used to report on the organization of a data base in terms of chain lengths and free space availability in order to indicate which portions of a data base require reorganization.

6. **Partial Data Base Reorganization Utility (DFSPRCT1):** used to reorganize selected parts of HDAM or HIDAM data bases as indicated by the output of the Data Base Surveyor.

7. **Data Base Prereorganization (DFSURPR0):** used to create a control data set which contains information on logical relationships and which serves as input to the following three utilities.

8. **Data Base Scan (DFSURGS0):** used to extract information on logical relationships from data bases which are not being reorganized and to create a data set containing this information for input to the Prefix Resolution utility.

9. **Data Base Prefix Resolution (DFSURG10):** used to create a data set that contains the prefix information needed to update the pointers for segments which are used in logical relationships.

10. **Data Base Prefix Update (DFSURGP0):** used to perform the actual prefix pointer updates indicated by the output of the Prefix Resolution utility.

The first six of the programs are known as the **Physical Reorganization Utilities** and the last four are known as the **Logical Relationship Resolution Utilities**. A complete descriptions of these utilities may be found in the **IMS/VS UTILITIES REFERENCE MANUAL, SH20-9029**

The Physical Reorganization Utilities, except for the Data Base Surveyor, perform two basic functions, unload and reload. Unload is done segment by segment in hierarchical sequence through a series of GET-NEXT calls. If the data base is badly disorganized or very large, the unload may take an extremely long time. Reload is also done in hierarchical sequence and is effectively an initial load of the data base. The time required for reloading the data base will be approximately the same as the time for the initial load of that data base.

The Primary Index of an HIDAM data base will be completely rebuilt by Reload but, if any Secondary Indexes are present, the Logical Relationship Resolution Utilities must also be used to complete their reconstruction. Should the data base being reorganized participate in any logical relations, then the Logical Relationship Resolution Utilities must be used in order to ensure that all of the prefix pointers are correct.

7.2.2.2 Re-org Capabilities of VSAM

Although both REPRO and EXPORT/IMPORT reorganize VSAM data sets, this is **not** equivalent to data base reorganization. VSAM reorganization works in terms of **logical records** as opposed to data base reorganization which works in terms of **segments**. All that VSAM reorganization is able to do is to manipulate the structure and sequence of the logical records. VSAM is not aware of the data base segments stored in its logical records and is unable to reorder them into hierarchical sequence. Furthermore, VSAM has no ability to produce the work data sets which are required for reorganization of data bases with logical relationships. If logical relationships exist for the data base being reorganized, the IMS reorganization utilities **must** be used.

When a KSDS is reorganized, the records are loaded in key sequence, VSAM free space is reallocated according to the values specified in the cluster definition and the index component is rebuilt. This removes the fragmentation of the data set caused by CI and CA splits and recovers free space which had been used by insertions.

Reorganization of an ESDS must involve a user program since simply unloading and then reloading the data set will not cause any change in the structure or sequencing of an ESDS. An ESDS can be effectively reorganized by a user written program which creates a data set consisting of those records which are still current, dropping any records which may have been flagged or otherwise indicated as being 'deleted'. The ESDS may then be emptied and the data set created by the user program REPROed into it. This is the only means available to restructure an ESDS and to reclaim space occupied by unwanted records since delete (ERASE) requests are not allowed for an ESDS.

7.2.2.3 Access Method Services as a Reorganization Tool

Although the REPRO and EXPORT/IMPORT commands are unable to deal with data base segments or logical relations, there are certain circumstances under which these commands can be effectively used to reorganize data bases. The applicability of Access Method Services as a means of reorganization is limited to the KSDS components of HISAM, SHISAM and INDEX data bases. Access Method Services is not suitable to the reorganization of ESDS since this requires the intervention of some sort of special unload program.

In SHISAM and INDEX KSDS, the segments are each stored in a separate logical record and identified by unique keys so that there is no real difference between VSAM key sequence reordering and IMS hierarchical sequence reordering. For the case of an HISAM KSDS, there may be multiple segments in each logical record but these segments are continually maintained in hierarchical sequence within the record. Thus, VSAM reorganization will achieve the same result as will the IMS utilities with the single exception that VSAM reorganization will not eliminate deleted segments from the data base.

It is recommended that REPRO be used in reorganization rather than EXPORT/IMPORT for the following reasons:

- Statistical information in the catalog is used to monitor the status of data sets and to determine the need for reorganization. It should therefore be reset with each reorganization in order to provide the current rather than the historical status of the data set.

- Avoiding the extra overhead of copying and restoring catalog entries will decrease the elapsed time for reorganization and allow the data base to be restored to a usable condition sooner.

- The ability to selectively copy and restore portions of the data base with REPRO provides a partial reorganization capability. This can be very useful in cases where fragmentation of the data set is localized and it is necessary to return the data base to operational status as soon as possible.

Reorganization with REPRO is done in three steps:

1. Make a copy of the KSDS by REPROing it to a sequential data set on tape or disk.

2. Empty the data set by deleting and then redefining it. Alternatively, the REUSE or REPLACE options can be used in step 3 and this step may then be omitted.

3. Create the reorganized data set by REPROing the copy made in step 1 back into the KSDS.

If Backup has been taken with EXPORT and it is desired that the data set be both restored and reorganized, then this may be accomplished by simply IMPORTing the Backup copy.

The use of Access Method Services to reorganize IMS/VS data bases is suitable for SHISAM, HISAM, HIDAM Primary Indices and Secondary Indices with unique keys which have experienced a high level of insert activity and which do not participate in logical relationships. It is not necessary to make any special provisions for the Dependent Overflow pointers in HISAM data bases nor for the prefix pointers in INDEX data bases during reorganization with Access Method Services. This is because these pointers are not changed by reorganization of the data bases which contain them.

Access Method Services should **not** be used for HISAM data bases which have logical relationships or which are being reorganized in their entirety to reclaim space occupied by deleted segments. Nor should it be used for INDEX data bases when the data base on which the index is defined is also being reorganized.

It is likely that Access Method Services reorganization will be faster than the IMS Physical Reorganization Utilities in the areas where it is applicable but it is the users' responsibility to evaluate the suitability of this technique to their own environment.

7.2.3 VERIFY

There is a macro which is part of the VSAM system called VERIFY. This is the same macro which is issued internally by the Access Method Services command VERIFY.

The purpose of VERIFY is to ensure that the catalog value which contains the highest used RBA (HURBA) of the data set component is not **less** than the formatted Software-end-of-file (SEOF) control interval (last 4 bytes of the CI are binary zeroes).

VERIFY validates the HURBA by accessing the data or index control interval pointed to by the corresponding current HURBA indicated in the catalog. This control interval is compared for a SEOF control interval. If it is not recognized as an SEOF control interval, the control intervals are accessed **forward** from that point until the SEOF is found or until the high allocated RBA is reached. If the SEOF is lower than the HURBA, unpredictable results will be encountered (i.e. I/O error, etc.).

IMS/VS internally issues the VERIFY macro. This is performed in batch utilities and user programs as well as in the online system whenever required. Beginning with IMS/VS 1.1.5, this is also performed by batch IMAGE COPY. VERIFY is performed even in cases where not absolutely necessary, but is done as a safeguard. Therefore, an IMS/VS program accessing a VSAM DL/I data base need not include an Access Method Services VERIFY jobstream immediately prior or subsequent to the execution of the IMS/VS program. The only case may be in SHISAM when the DL/I file is OPENed as a native VSAM data set and accessed with non-DL/I requests.

8.1 VSAM CATALOG INFORMATION

Catalog records for a VSAM data set contain the following information:

- Device type and volume serial numbers of volumes containing the data set.
- Location of the extents of the data set and secondary allocations, if any.
- Attributes of the data set, such as control interval size, physical record size, number of control intervals in a control area, location of the primary key field for a KSDS, etc.
- Statistics, such as the number of insertions, updates, retrievals, splits, etc.
- Password protection information.
- An indication of the connection between data sets and their index(es), for example, the index and data components of a KSDS.
- Historic information, such as creation and expiration dates, owner identification, etc.

A VSAM catalog also contains information regarding the location of data spaces and available space on volumes that contain VSAM data sets. The information in the VSAM catalog is very important for monitoring the status of the data sets. It can be used to determine if the data set is extending more rapidly than is expected and requires extra space allocation, if space requirement calculations were accurate and if the level of activity in the data set has caused fragmentation necessitating reorganization.

The following catalog listings and descriptions have been extracted from the manual **VSAM PRIMER AND REFERENCE, G320-5774,** which contains more detailed descriptions.

8.1.1 Allocation Information

Each VSAM ESDS has allocation information for its DATA component, and a KSDS for its DATA component and its INDEX component. All catalog information can be obtained by using the **LISTCAT** command of the Access Method Services utility.

8.1.1.1 CLUSTER Allocation

The cluster has no allocation information. The allocation information for a cluster is included in the DATA component (ESDS), and in the DATA and INDEX components for a KSDS.

8.1.1.2 DATA Component Allocation

The following is an example for the DATA component of data set (cluster) CUSTOMER.K (the LISTCAT output had to be modified to fit on the page):

```
CLUSTER - CUSTOMER.K
 DATA - CUSTOMER.K.D
  ALLOCATION
   SPACE-TYPE---CYLINDER HI-ALLOC-RBA--1013760
   SPACE-PRI-----------5 HI-USED-RBA---1013760
   SPACE-SEC-----------2
  VOLUME
   VOLSER---------WTVSAM PHYREC-SIZE------1024 HI-ALLOC-RBA--1013760
   DEVTYPE---X'30502009' PHYRECS/TRK--------11 HI-USED-RBA---1013760
   VOLFLAG---------PRIME TRACKS/CA----------19
   EXTENTS:
   LOW-CCHH--X'00040000' LOW-RBA-------------0 TRACKS-------------95
   HIGH-CCHH-X'00080012' HIGH-RBA------1013759
```

Figure 43. LISTCAT output: DATA component allocation information

Explanation:

SPACE-TYPE	the allocation of this data component is in terms of CYLINDERs.
SPACE-PRI	the primary allocation value is 5 cylinders.
SPACE-SEC	the secondary allocation value is 2 cylinders.
HI-ALLOC-RBA[20]	the highest RBA (plus 1) allocated by VSAM for this data component to store data. Points to the end of the last extent currently allocated.
HI-USED-RBA[20]	the highest RBA (plus 1) within the last CA that actually contains data (in KSDS it is the highest RBA in the last CA containing data). If this value is 0, the data set is considered to be empty.
VOLSER	the volume serial number WTVSAM.
DEVTYPE	the device type is 3330 (see Access Method Services manual).
VOLFLAG	PRIME indicates that this is the first volume on which data records for this data component are stored.
PHYREC-SIZE	physical record size.
PHYSRECS/TRK	11 physical records with 1024 bytes are written per track on a 3330.

---

[20] HI-ALLOC-RBA and HI-USED-RBA are included both under ALLOCATION and VOLUME. The ALLOCATION part gives overall information for the volume. If there are multiple keyranges on a volume multiple VOLUME entries will exist (one per keyrange) each with HI-ALLOC-RBA and HI-USED-RBA per keyrange.

TRACKS/CA	one CA is 19 tracks. Since the allocation was made in terms of CYLINDERs, VSAM set the CA size equal to one cylinder. One cylinder on a 3330 contains 19 tracks.
LOW-CCHH	the starting physical address of this extent of the data set. The first four digits are the cylinder number in hexadecimal and the last four digits are the track number in hexadecimal.
HIGH-CCHH	the ending physical address of this extent of the data set in the same format as for LOW-CCHH.
LOW-RBA	the lowest RBA in this extent of the data set. 0 indicates that this is the first extent.
HIGH-RBA	the highest RBA in this extent.
TRACKS	the total number of tracks in this extent of the data set.

This information is very useful in locating the data set on disk and in determining whether the data set has taken secondary extents (the EXTENT information could not be shown here due to limited page width).

8.1.1.3 INDEX Component Allocation

```
CLUSTER ------- CUSTOMER.K
 INDEX ------ CUSTOMER.K.I
  VOLUME
   VOLSER---------WTVSAM PHYREC-SIZE------2048 HI-ALLOC-RBA----12288
   DEVTYPE---X'30502009' PHYRECS/TRK---------6 HI-USED-RBA------2048
   VOLFLAG---------PRIME TRACKS/CA-----------1
   EXTENTS:
   LOW-CCHH--X'00030011' LOW-RBA-------------0 TRACKS--------------1
   HIGH-CCHH-X'00030011' HIGH-RBA--------12287
  VOLUME
   VOLSER---------WTVSAM PHYREC-SIZE------2048 HI-ALLOC-RBA----22528
   DEVTYPE---X'30502009' PHYRECS/TRK---------6 HI-USED-RBA-----22528
   VOLFLAG---------PRIME TRACKS/CA----------19
   EXTENTS:
   LOW-CCHH--X'00040000' LOW-RBA---------12288 TRACKS-------------95
   HIGH-CCHH-X'00080012' HIGH-RBA--------22527
```

Figure 44. LISTCAT output: INDEX component allocation information

IMBED was specified for the cluster, therefore, two extents are shown. The first extent contains only the high-level index (1 track).

The second extent contains the imbedded index sequence set and the DATA component. The same extent is also shown when listing the DATA component (see previous section).

8.1.2 Attributes Information

8.1.2.1 Cluster Attributes

The CLUSTER component **HISTORY** and **PROTECTION** group is similar to the same groups in the DATA component, which is explained in the next section. The **ASSOCIATIONS** group contains the DATA and INDEX component names of this cluster.

8.1.2.2 DATA Component Attributes

The following is one part of the LISTCAT output of the DATA component (output is concentrated to fit on this page):

```
DATA ------- CUSTOMER.K.D
HISTORY
 OWNER-ID.-(NULL) CREATION--79.355 RCVY-VOL-WTVSAM RCVY-CI-X'00000D'
 RELEASE--------2 EXPIRAT.--00.000 RCVY-DEVT--X'30502009'
PROT.PSWD--(NULL) RACF--------(NO)
ASSOCIATIONS
  CLUSTER--CUSTOMER.K
ATTRIBUTES
 KEYLEN---------5 AVGLRECL-----200 BUFSPACE---4096 CISIZE-------1024
 RKP------------0 MAXLRECL-----200 EXCPEXIT-(NULL) CI/CA---------198
SHROPTNS(1,3)  RECOVERY  SUBALLOC  NOERASE  INDEXED  NOWRITECHK
IMBED NOREPLICAT  UNORDERED  NOREUSE  NONSPANNED
```

Figure 45. LISTCAT output: DATA component history and attributes

The HISTORY group contains the following parameters:

OWNER-IDENT Identifies the OWNER (owner-id) of the object. '(NULL)' means no owner-id was specified.

CREATION the Julian date (YY.DDD) the entry was created.

EXPIRATION the cluster can be deleted without specifying the PURGE parameter (00.000).

RCVY-VOL Used for recoverable catalogs. Gives 'volser' of CRA volume which contains duplex catalog information.

RCVY-CI Used for recoverable catalogs (CRA CI number).

RELEASE Release number of VSAM under which the entry was created (all current OS/VS systems use VSAM Release 2).

RECY-DEVT Used for recoverable catalogs (device type = 3330). Gives device type of CRA volume.

The PROTECTION group contains the following entries:

PROTECTION-PSWD indicates the MASTERPW, CONTROLPW, UPDATEPW and/or READPW for this object. '(NULL)' means no password has been specified.

RACF (MVS only) '(NO)' means this entry is not RACF protected.

The ASSOCIATIONS group lists the types and entry names of each entry associated with the present entry (no entry is associated to the DATA component).

The ATTRIBUTES group contains the following entries:

KEYLEN length of key (5 bytes).

RKP relative key position in the logical record (key starts with byte 0).

AVGLRECL logical record length average is 200 bytes.

MAXLRECL maximum record length is 200 bytes.

BUFSPACE minimum space in bytes that will be used by VSAM for data and index buffers for this KSDS cluster.

CISIZE control interval size for this data component in bytes.

CI/CA number of control intervals per control area.

EXCPEXIT '(NULL)' means no exception exit routine entry was specified in the DEFINE command.

SHROPTNS the SHAREOPTIONS values are '1' for **cross-region** and '3' for **cross-system**.

RECOVERY is the loading option. The opposite attribute is SPEED.

SUBALLOC the space for this cluster is suballocated by VSAM from a suballocatable data space.

NOERASE the data are not overwritten with X'00' when the entry is deleted.

INDEXED INDEXED identifies this cluster as KSDS.

NOWRITECHECK write operations are not checked for correctness. Specification of WRITECHECK results in time consuming operations not needed for DASD devices like IBM 3330, IBM 3340, or IBM 3350.

IMBED the sequence set index record is stored along with its data control area and is replicated on its track.

NOREPLICAT the index records on the higher index level are **not** replicated.

UNORDERED if more volumes are used, it specifies the order of using the volumes for allocation.

NOREUSE this data set cannot be used as work data set.

NONSPANNED the records cannot span control intervals.

Note that 3850 (MSS) default parameters NODESTAGEWAIT and STAGE are **not** listed with LISTCAT.

8.1.2.3 INDEX Component Attributes

```
INDEX ------ CUSTOMER.K.I
 HISTORY
  OWNER-ID.-(NULL) CREATION-77.355 RCVY-VOL-WTVSAM RCVY-CI-X'00000F'
  RELEASE--------2 EXPIRATION---00.000 RCVY-DEVT-X'30502009'
 PROTECTION-PSWD---(NULL) RACF----(NO)
 ASSOCIATIONS
   CLUSTER--CUSTOMER.K
 ATTRIBUTES
  KEYLEN---------5 AVGLRECL------0 BUFSPACE-------0 CISIZE------2048
  RKP------------0 MAXLRECL---2041 EXCPEXIT--(NULL) CI/CA----------6
  SHROPTNS(1,3)    RECOVERY    SUBALLOC    NOERASE    NOWRITECHK
  IMBED    NOREPLICAT    UNORDERED    NOREUSE
```

Figure 46. LISTCAT output: INDEX component history and associations

The INDEX component **HISTORY**, **PROTECTION**, **ASSOCIATIONS**, and **ATTRIBUTES** groups are similar to the same groups in the DATA component, which were explained in the previous section.

Note that the **BUFSPACE** value for the index component is 0. This is because the buffer requirements of the index are included in the BUFSPACE for the data component.

8.1.3 Statistics

VSAM keeps statistics for each component of a data set.

This information is updated in the catalog if the data set is closed, or if the data set allocates a new extent.

8.1.3.1 Cluster Statistics

The CLUSTER component has no **STATISTICS** group.

8.1.3.2 DATA Component Statistics

The DATA component **STATISTICS** group is printed as follows:

```
STATISTICS
 REC-TOTAL--------3000 SPLITS-CI-----------0 EXCPS-------------1601
 REC-DELETED---------0 SPLITS-CA-----------0 EXTENTS--------------1
 REC-INSERTED--------0 FREESPACE-%CI------20 SYSTEM-TIMESTAMP:
 REC-UPDATED---------0 FREESPACE-%CA------10    X'8DDE7F92DF1F7800'
 REC-RETRIEVED----3010 FREESPC-BYTES--245760
```

Figure 47. LISTCAT output: DATA component statistics

Explanation:

REC-TOTAL	this data component contains 3000 records. This is the **only** field being updated during initial load.
REC-DELETED	no records have been deleted since the data set was loaded.
REC-INSERTED	no records have been inserted after loading the data set (records added at the end are not counted here, therefore this field will always be '0' for an ESDS).
REC-UPDATED	no records have been updated in the data set. This field is updated when a PUT UPDATE request is issued for a record.
REC-RETRIEVED	3010 records have been read.
SPLITS CI/CA	number of CI and CA splits since the data set was loaded. **The 'SPLITS-CA' field should be examined frequently.** If the value increases too much, the data set should be reorganized.
FREESPACE-%CI/CA	20% free space in the CI (1024 * 20% = 204 bytes) and 10% of the number of CI per CA (198 * 10% = 19 CI) are left free when the data set is loaded.
FREESPC-BYTES	this is the amount of free bytes in the data component, excluding the free space in the partially filled CI's. This value is calculated by multiplying the number of free (never used) CIs by the CI size.
EXCPS[21]	1601 EXCPs have been issued against this data component.
EXTENTS	the number of extents of this data component.
SYSTEM-TIMESTAMP	the time (System/370 time-of-day clock value) this data component was last closed (no important timestamp).

[21] These are not really EXCPs. VS1=EXCPVR, SVS=EXCPVR, MVS=STARTIO interface to IOS. A GTF trace of VSAM I/O will not show any EXCPs.

8.1.3.3 INDEX Component Statistics

The INDEX component STATISTICS group is printed as follows:

```
STATISTICS
 REC-TOTAL----6 SPLITS-CI-----0 EXCPS------------48 INDEX:
 REC-DELETED--0 SPLITS-CA-----0 EXTENTS-----------2 LEVELS----------2
 REC-INSERTED-0 FREESPACE-%CI-0 SYSTEM-TIMESTAMP:   ENTRIES/SECT---14
 REC-UPDATED--0 FREESPACE-%CA-0 X'8BDE7F92DF1F7800' SEQ-SET-RBA-12288
 REC-RETRIEV.-0 FREESPC-BYTES--10240                HI-LEVEL-RBA----0
```

Figure 48. LISTCAT output: INDEX component statistics

Most fields are identical with the same fields in the STATISTIC group of the DATA component and were explained in the previous section.

The following fields are important for the INDEX component:

INDEX LEVELS '2' specifies that 2 index levels are built. The first level is the 'Sequence set level' which is stored with the data component (IMBED). The second level is the 'Index set level'.

REC-TOTAL '6' means a total of 6 index records (CIs) exist. The sequence set consists of 5 index records (one per CA) and the index set consists of one index record (with the five entries pointing to the sequence set records).

Note that the number of extents in the index component is 2, one for the index set and one for the sequence set.

An index component which has the IMBED option will always have at least one more extent than does the data component.

8.2 DATA BASE STATISTICS

In addition to the information about data sets found in the VSAM catalog, there is another important source of information on the behavior of data bases consisting of VSAM data sets. This is the statistical data which is maintained by the system during execution. There are two separate sets of statistics, the **Buffer Handler Statistics** and the **VSAM Statistics**.

8.2.1 Source of Statistical Information

The Buffer Handler Statistics are kept in a special control block in the DL/I Buffer Handler pool and are maintained by the IMS to VSAM interface routine, DFSDVSM0.

VSAM statistics are kept in the Header control block (BSPH) of each subpool of VSAM buffers and are maintained by the VSAM shared resources facility (both GSR and LSR). The VSAM statistics are moved into the DL/I Buffer Handler Pool for reporting purposes when they are edited by a statistics (STAT) call, an operator issued DISPLAY command or one of the Monitor programs.

Output from the STAT call and DISPLAY command is similar to that of the Monitors and, since this section is concerned with the reports produced from the Monitor programs, there will be no further discussion of STAT or DISPLAY (for information on the STAT call refer to the **IMS/VS APPLICATION PROGRAMMING REFERENCE MANUAL, SH20-9026** and for information on the DISPLAY command refer to the **IMS/VS OPERATOR'S REFERENCE MANUAL, SH20-9028.**)

8.2.2 The Monitor Programs

IMS/VS provides two separate monitors, the **DB Monitor** and the **DC Monitor**. The function of each monitor is to gather and format IMS/VS performance related data and to record this data on a statistics log. Information on how to operate these monitors is found in the sections entitled 'Using the IMS/VS DB Monitor' and 'Using the IMS/VS DC Monitor' in the **IMS/VS OPERATOR'S REFERENCE MANUAL, SH20-9028.**

The DB Monitor (DFSMNTB0) is available in IMS/VS DB (batch) systems. It can monitor the activity between application programs and data bases.

The DC Monitor (DFSMNTR0) is available to IMS/VS DB/DC (online) systems. In addition to performing all of the functions of the DB Monitor, it can track and record information about activities occurring in the IMS/VS control region as well as data communication activities.

The statistics produced by each monitor are used as input to the related report programs.

8.2.3 Monitor Report Printing Programs

The monitor report printing programs are offline utilities that organize and print the performance related information collected by the monitors. These reports summarize and categorize the traced IMS/VS activities at various levels of detail. Each monitor run will produce enough data for the offline utilities to produce a number of different reports. These reports may contain totals, averages, and user defined or default distribution displays for the activities reported.

The DB Monitor Report Printing Program is DFSUTR30. It can produce the following reports:

- Statistics from Data Base Buffer Pools and VSAM Buffer Subpools
- Program I/O
- DL/I Call Summary
- VSAM Statistics (only available from DFSUTR30)
- Monitor Overhead
- Distribution Appendix

The accuracy of these reports depends upon the accuracy of the records in the data set produced by the DB monitor. Records of all events are expected in pairs, start-event and end-event; events are not counted or reported unless both records are received.

The DC Monitor Report Printing Program is DFSUTR20. The report formats and the nature of the information in the reports produced by DFSUTR20 are identical or similar to those produced by DFSUTR30. The following types of reports are printed by DFSUTR20:

- System Configuration
- Statistics from Buffer Pools
- Region Summary and IWAIT
- Program Timing, Calls, IWAITs, Scheduling and Dequeueing
- Communication Traffic and Timing
- Transaction Queueing
- Dl/I Call Summary
- General Reports
- Run Profile
- Distribution Appendix

Like DFSUTR30, the accuracy of the reports produced by DFSUTR20 depends on that of the corresponding monitor program.

8.2.4 VSAM Related Reports

Of all the reports produced by DFSUTR30 and DFSUTR20, there are only two of specific interest in terms of VSAM related information. These are the VSAM Buffer Subpools Report, which is printed by both programs, and the VSAM Statistics Report, which is only printed by DFSUTR30.

Examples of these reports are shown below. The values shown in these examples are not intended to reflect actual values that may be received by a user's execution of the utilities.

8.2.4.1 VSAM Buffer Subpools Report

The following is an example of the VSAM Buffer Subpools Report (the format of this report has been condensed and modified slightly in order to make it fit on to the page).

```
IMS MONITOR **BUFFER POOL STATISTICS** START 13:24:10  STOP 13:30:16
    V S A M   B U F F E R   P O O L    SUBPOOL ID              :    1
                                       SUBPOOL BUFFER SIZE[22] : 2048
                                       TOTAL BUFFERS IN POOL[22]:   20

                                                 START  STOP  DIFF
                                                 TRACE TRACE
NUMBER OF RETRIEVE BY RBA CALLS[22]                  0    503   503
NUMBER OF RETRIEVE BY KEY CALLS[22]                  0     73    73
NUMBER OF LOGICAL RECORDS INSERTED INTO ESDS[22]     0      0     0
NUMBER OF LOGICAL RECORDS INSERTED INTO KSDS[22]     0     13    13
NUMBER OF LOGICAL RECORDS ALTERED IN THIS SUBPOOL[22] 0    179   179
NUMBER OF TIMES BACKGROUND WRITE FUNCTION INVOKED[22] 0      0     0
NUMBER OF SYNCHRONIZATION CALLS RECEIVED[22]         0      0     0
NUMBER OF VSAM GET CALLS ISSUED[22]                  0    249   249
NUMBER OF VSAM SCHBFR CALLS ISSUED[22]               0     21    21
NUMBER OF TIMES CI REQUESTED ALREADY IN POOL[23]     0    357   357
NUMBER OF CONT INT READ FROM EXTERNAL STORAGE[23]    0      9     9
NUMBER OF VSAM WRITES INITIATED BY IMS/VS[23]        0      0     0
NUMBER OF VSAM WRITES TO MAKE SPACE IN POOL[23]      0      0     0
NUMBER OF PERM WRT ERROR BUFFS NOW IN SUBPOOL[23]    0      0     0
LARGEST NUMB OF PERM ERR BUFFS EVER IN SUBPOOL[23]   0      0     0
```

Figure 49. Statistics from VSAM Buffer Subpools Report

The header information shows the time and date (date not shown because of lack of space) at which the monitor began and ended tracing of activity as well as the subpool identification number, buffer size and number of buffers. A report in this format will be produced for each buffer subpool defined by the user in the DFSVSAMP data set or the DFSVSMxx member of IMSVS.PROCLIB.

---

[22] Values derived from DL/I Buffer Handler Statistics.

[23] Values derived from VSAM statistics (BUFH).

The values for the various entries are shown for the start of the trace and the end of the trace (START and STOP columns). Changes in these values during the period of the trace are shown in the DIFFERENCE column.

Explanation of the entries:

RETRIEVE BY RBA	number of accesses by RBA, usually to an ESDS or as a result of following a pointer.
RETRIEVE BY KEY	number of accesses by key to HISAM or HIDAM root segments or to a Secondary Index.
INSERT IN ESDS	number of times insertion of a root or dependent segment required a new logical record to contain the new segment in an HDAM, HIDAM or HISAM data base or a shifted segment needed a new logical record in an HISAM data base.
INSERT IN KSDS	number of new root segments inserted into an HISAM or INDEX data base.
ALTERED	number of logical records updated as a result of REPL or DLET calls.
BACKGROUND WRITE	number of times the Background Write facility (specified in the OPTIONS statement) was invoked.
SYNCH CALLS	number of times an application program reached a synchronization point.
VSAM GETS	self explanatory
VSAM SCHBFR	number of times IMS requested VSAM to search the buffer subpool to determine whether or not the required CI is already in a buffer.
CI IN POOL	number of times IMS found the required segment in a CI already in the buffer subpool.
READ FROM EXTERNAL	number of times a CI had to be read from external storage to satisfy a request.
WRITES BY IMS	number of times IMS rather than VSAM requested a CI to be written to external storage.
WRITES FOR SPACE	number of times a CI had to written out to free a buffer for reuse.
WRT ERRORS NOW	number of buffers currently containing a CI that cannot be written to DASD because of some sort of write error. These buffer are not reused until the write is successful.
WRT ERRORS EVER	largest number of buffers held because of write errors.

8.2.4.2 VSAM Statistics Report

The following is an example of the VSAM statistics report produced by DFSUTR30.

The format of the report has been condensed and modified in order to make it fit on the page. Four columns of values have had to be eliminated from the report because of space limitation. These columns are headed: FOUND, READS, USR WTS, and NUR WTS and follow immediately after the SCHBFR column.

IMS MONITOR **VSAM STATISTICS** START 13:24:10 STOP 13:30:16

PCBNAM	DDNAM	DL/I FUNC	VSAM CALL	RET RBA	RET KEY	ISRT ESDS	ISRT KSDS	BFR ALT	BKG WTS	SYN PTS	GETS	SCHBFR
CHVBT2	DHSK09	ISRT	114	1.71	0.00	0.00	0.00	0.93	0.00	0.00	0.75	0.26
		GN	10	9.40	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
		GU	39	2.10	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
		DLET	80	2.67	0.00	0.00	0.00	1.30	0.00	0.00	0.70	0.00
	DD TOTAL		243	2.41	0.00	0.00	0.00	0.88	0.00	0.00	0.98	0.12
	DXSK01	ISRT	54	0.11	0.85	0.00	0.43	0.18	0.00	0.00	1.33	0.00
		GN	5	0.80	2.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
		GU	34	0.47	2.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
		DLET	22	4.27	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00
	DD TOTAL		115	1.04	1.26	0.00	0.22	0.27	0.00	0.00	1.49	0.00
	DXSK09	ISRT	49	2.69	0.00	0.00	0.00	1.10	0.00	0.00	0.65	0.24
		GU	9	2.66	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
		GN	5	4.40	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
		DLET	37	3.02	0.00	0.00	0.00	1.40	0.00	0.00	0.59	0.00
	DD TOTAL		102	2.90	0.00	0.00	0.00	1.07	0.00	0.00	0.80	0.11
	DHSK04	GU	2	2.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
	DD TOTAL		2	2.00	0.00	0.00	0.00	0.00	0.00	0.00	2.00	0.00
PCB TOTAL			462	2.17	0.31	0.00	0.05	0.77	0.00	0.00	1.07	0.09
BATCH TOTAL			462	2.17	0.31	0.00	0.05	0.77	0.00	0.00	1.07	0.09

Figure 50. VSAM Statistics Report

This report provides statistics on a **per call** basis for changes in 13 subpool statistical values between the start and end of the trace. A set of statistics is provided for each combination of program (PCBNAME), call function (DL/I FUNC) and data set (DDNAME) detected during the trace. An occurrence count is provided for each combination. Each set of statistics is a summation of the changes in **all** the subpools divided by the number of occurrences.

Summary lines show totals for each program (PCB), for each data set referenced by the program (DD NAME under PCB), and for the complete duration of the trace period (BATCH).

The statistical values reported on are the same as those shown in the VSAM Buffer Subpools Report.

Explanation of the headings:

RET RBA number of retrieves by RBA.

RET KEY number of retrieves by key.

ISRT ESDS number of logical records inserted into ESDSs.

ISRT KSDS number of logical records inserted into KSDSs.

BFR ALT number of logical records altered in the buffers.

BKG WRT number of times the Background Write facility was invoked.

SYN PTS number of synchronization calls received by the buffer handler.

GETS number of VSAM GET calls issued.

SCHBFR number of VSAM SCHBFR calls issued.

FOUND (not shown) number of times the requested CI was found to be already in the buffers.

READS (not shown) number of times VSAM read a CI from external storage.

USR WTS (not shown) number of VSAM writes initiated by IMS/VS.

NUR WTS (not shown) number of VSAM writes issued to make space in the subpool.

ACB Access Method Control Block (VSAM DCB equivalent)
CA Control Area
CI Control Interval
CIDF Control Interval Definition Field
CNV Control Interval (used for control interval access)
CRA Catalog Recovery Area
CSA Common Service Area
DASD Direct Access Storage Device
DBD Data Base Definition
DCB Data Control Block (OS/VS control block to describe a nonVSAM data set)
DDEP Direct Dependent segment
DEDB Data Entry Data Base
DSCB Data Set Control Block (in the VTOC)
EMH Expedited Message Handler
ESDS Entry Sequenced Data Set
FBFF Free Block Frequency Factor
FP Fast Path Feature
FSAP Free Space Anchor Point
FSPF Free Space Percentage Factor
GSAM Generalized Sequential Access Method
GSR Global Shared Resources
HISAM Hierarchical Indexed Sequential Access Method
HSAM Hierarchical Sequential Access Method
HDAM Hierarchical Direct Access Method
HIDAM Hierarchical Indexed Direct Access Method
HURBA Highest-used-RBA
IC IMAGE COPY Utility
ICI Improved Control Interval access
ICIP Improved Control Interval Processing
ILP Initial Load Program (User)
IMS/VS Information Management System/Virtual Storage
IOB Input Output Block
IS Index Set (part of the index)
KSDS Key Sequenced Data Set
LSR Local Shared Resources
LTWA Log Tape Write Ahead
MSDB Main Storage Data Base
NSR Nonshared Resources
OSAM Overflow Sequential Access Method
PLH Placeholder (VSAM control block)
MCAT VSAM Master Catalog
RAP Root Anchor Point
RBA Relative Byte Address
RDF Record Definition Field
RRDS Relative Record Data Set
SDEP Sequential Dependent segment
SEOF Software-end-of-file
SIP Split-In-Progress Flag (bit)
SS Sequence Set (part of the index)
SHISAM Simple Hierarchical Indexed Sequential Access Method
SHSAM Simple Hierarchical Sequential Access Method
UCAT VSAM User Catalog
VSAM Virtual Storage Access Method
UOW Unit-of-Work (see Fast Path)
VTOC Volume Table Of Contents (disk directory)

A

B

C

D

I

K

L

M

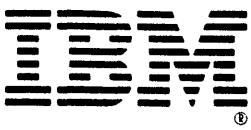
N

Reader's Comment Form

Fold

IBM World Trade Corporation
World Trade Systems Center
Department 471, Building F27
555 Bailey Avenue
P.O. Box 50020
San Jose, California 95150
U.S.A.

Fold



International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas / Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe / Middle East / Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

Your comments please.....

Reply requested

Yes ☐

No ☐

Name_____

Job Title_____

Address_____

Thank you for your cooperation

Reader's Comment Form

Fold

IBM World Trade Corporation
World Trade Systems Center
Department 471, Building F27
555 Bailey Avenue
P.O. Box 50020
San Jose, California 95150
U.S.A.

Fold

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas / Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe / Middle East / Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

IBM

Palo Alto Systems Center

Technical Bulletin

IMS/VS AND OS/VS VSAM BUFFER OPTIONS (SHARED VS. NON-SHARED RESOURCES)

by Wayne Weikel, Edward H. Daray
Palo Alto Systems Center

G320-6035
April, 1979

TABLE OF CONTENTS

1.0 INTRODUCTION

The purpose of this technical bulletin is to discuss the VSAM parameters associated with buffering techniques that the user may specify when executing IMS/VS user-written programs and IMS/VS utility programs, and the effect of the parameter specifications upon performance.

The information applies to the IMS/VS 1.1.5 system. The material is generally applicable to releases prior to IMS/VS 1.1.5. There are some instances when the implementation differs slightly on pre-IMS/VS 1.1.5 releases and no attempt has been made in this document to note these differences.

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "As Is" basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The information presented should be used only as a guide. The implementation of each feature discussed may vary in a given system environment and should be evaluated relative to its applicability and the requirements of the installation.

2.0 VSAM PARAMETERS RELATED TO BUFFERING TECHNIQUES

The Virtual Storage Access Method (VSAM) has two options for allocating and managing the buffers to be used for input and output operations to a VSAM data set. These options are selected by the user issuing the OPEN request. One option is referred to as a non-shared resources (NSR) and the IMS/VS community commonly refers to this as "native VSAM". With this option, the buffers and I/O-related control blocks are allocated for use by a single data set. Each data set OPEN'ed with this option has a set of these buffers and control blocks built which are used by requests directed to that data set. The other option for buffer management is referred to as locally or globally shared resources (LSR/ GSR). With this implementation, these buffers and I/O related control blocks become part of a resource pool that can be shared by multiple data sets. With this option, a pool of resources is first constructed and subsequent data set OPEN's which specify LSR/GSR will be directed to the pool in order to obtain buffers and I/O-related control blocks to perform input/output operations. Therefore, NSR has the connotation of data set related and LSR/GSR that of a pool shared by many data sets.

The significance of the prior discussion is that VSAM buffer options, specifiable by the IMS/VS user, fall into one of these two categories. The data set related options shown below as Figure 1 are only in effect when IMS/VS uses the NSR buffer management technique. These specifications are ignored by VSAM OPEN when a data set is directed to the pool (LSR/GSR) for these resources. It should also be noted that any options in the JCL "AMP" parameter override any previous definitions made through the ACB, GENCB macros. In the case of data set related buffers, the BUFND/BUFNI also override the BUFFERSPACE (BUFSP) value assigned during AMS DEFINE.

The other technique of buffer management, LSR/GSR, requires that the user build a pool of resources to be shared by all data sets OPEN'ed to the pool. IMS/VS builds this pool using the VSAM macro BLDVRP to the configuration specified by the user with the DFSVSAMP data set. This data set indicates the sub-pool sizes and the number of buffers within each of these sub-pools.

FIGURE 1. DATA SET RELATED BUFFER OPTIONS

PARAMETER	SOURCE	MEANING	RECOMMENDATIONS
BUFFERSPACE (BUFSP)	AMS define Cluster/Data	Minimum space to be allocated for buffers (both data and index) for use by this data set.	Do not specify. Allow to default = 2 X data CI size + 1 X index CI size.
BUFFERSPACE (BUFSP)		Maximum space to be allocated for buffers (both data and index) for use by this data set.	Do not specify. This can be provided by the BUFND/BUFNI parameters.
BUFND	Data set DD "AMP" parameter.	The number of data buffers to be allocated for use by this data set (Default = 2).	See discussion in 4.1 "Selecting a Value for BUFND"
BUFNI		The number of index buffers to be allocated for use by this data set (Default = 1).	Generally it is acceptable to allow this parameter to default. A slight performance improvement can be achieved in load/create mode by increasing this to a maximum value of the number of anticipated index levels.

3.0 IMS/VS UTILITIES FUNCTIONAL OVERVIEW

3.1 IMAGE COPY (DFSUDMP0)

The Image Copy utility is executed to obtain a backup copy of the IMS/VS data sets. This copy is usually written to tape.

The utility uses the native VSAM macros GET RPL or GETIX RPL to retrieve the VSAM logical records and writes these logical records to the output data set using QSAM. The GETIX is used when copying the VSAM index and such a copy is useful only for IMS/VS track recovery.

Immediately following a data set open, the VSAM VERIFY macro is issued. This should correct any problems caused by a previous close error. If the VERIFY fails, DFS2802 is issued and the job is terminated.

3.2 DATA BASE RECOVERY (DFSURDB0)

The Data Base Recovery utility is executed to assist in restoring a physically damaged data set within an IMS/VS data base, or it may be used to restore a data set to the conditions existing at some previous time.

The restoration process consists of restoring a backup copy and re-applying all changes since the copy was taken. This copy is usually an Image Copy.

Data base changes are recorded on IMS/VS log tapes. These log tapes can be processed by Change Accumulation which groups all changes by data set logical record. If multiple changes to the same data occur, only the latest change is retained. If multiple changes occur for the same logical record, but to different data, these changes are placed into one change accumulation record with the data offsets recorded.

Input to the recovery may be:

1) Image Copy (no changes are to be applied)

2) Image Copy and Change Accumulation (all changes have been processed by Change Accumulation)

3) Image Copy, Change Accumulation, and Log Tapes (some of the changes have been processed by Change Accumulation)

4) Image Copy and Log Tapes (no changes have been processed by Change Accumulation).

Because some users elect to obtain data set copies by other utilities (DASDR dump, etc.), three more possibilities exist. In this environment, the user first restores the data set from the backup copy then applies:

5) Change Accumulation (all changes have been processed by Change Accumulation)

6) Change Accumulation and Log Tapes (some changes have been processed by Change Accumulation)

7) Log Tapes (no changes have been processed by Change Accumulation)

The recovery will be performed in phases according to the user input. Phase 1 and Phase 2 are mutually exclusive. The valid input to each phase is:

Phase 1 Image Copy or Image Copy and Change Accumulation

Phase 2 Change Accumulation without an Image Copy

Phase 3 Log Tapes

During Phase 1, the Image Copy record is read, the Change Accumulation records are read, the change data overlays the Image Copy, then the logical record is written to the data base. This phase uses native VSAM PUT macros.

Phase 2 consists of randomly updating data set records using the Change Accumulation. Because the Change Accumulation records are sorted by RBA, this becomes a skip sequential form of processing. A call is made to the buffer handler to retrieve the logical record, the Change Accumulation record overlays the logical record using the contained offsets and lengths, and the logical record is written back using an Insert, Erase, or Buffer Alter call.

Phase 3 consists of randomly updating the partially restored data set with log tape input in the same fashion as Phase 2. This is totally random because of the log tape record sequence.

The user may substitute a HISAM Unload data set for the Image Copy. When this is done, certain processing changes occur. The HISAM unload does not contain a record for the ESDS CI zero, therefore, Recovery must create one. This is accomplished by a call to the buffer handler which will open the data set using NSR, write CI zero, close the data set, then open for output using GSR or LSR.

At the beginning of the recovery process, a test is made by the data base recovery program to insure that the data set being restored to is empty. This test results in the OS/VS VSAM OPEN routine issuing the following message:

IEC161I 072-053, Jobname, stepname, ddname, device address, volser, cluster name, data set name, catalog name.

This message should be regarded as normal and not an indication of an error condition.

3.3 DATA BASE BACKOUT (DFSBB000)

The Backout utility removes batch region changes made to a data base. All data base accesses are done using the buffer handler.

Backout is accomplished by reading the system log tape back-wards and reversing the data base changes. This is possible because the log tape contains both the before and after data base images.

On-line backout occurs when the IMS/VS online region issues a ROLL call or abends, and during Emergency Restart following a system failure. In these cases, the input is the system Dynamic Log which contains a copy of the records on the system log. The actual backout is performed by the same module as batch backout, therefore, all data base accesses use the buffer handler.

3.4 HISAM UNLOAD (DFSURUL0)

This utility is used to unload any HISAM or INDEX data base. All access to the data base is on a segment-by-segment basis via GET NEXT calls to the buffer handler.

The output data set becomes the input to the HISAM RELOAD utility, but optionally may be used in lieu of an Image Copy for Data Base Recovery.

3.5 HISAM RELOAD (DFSURRL0)

This utility is used to reload a HISAM data base unloaded by the HISAM UNLOAD utility. All access to the data base is on a segment-by-segment basis via INSERT calls to the buffer handler.

3.6 HD UNLOAD (DFSURGU0)

This utility is used to unload any HISAM, HIDAM, or HDAM data base. All access to the data base is on a segment-by-segment basis via DL/I GET NEXT calls.

The output data set is useable only as input to the HD RELOAD utility.

3.7 HD RELOAD (DFSURGL0)

This utility is used to reload a HISAM, HIDAM, or HDAM data base unloaded by the HD UNLOAD utility. All access to the data base is on a segment-by-segment basis using a special DL/I INSERT call.

4.0 RELATING VSAM BUFFER OPTIONS TO THE IMS/VS ENVIRONMENT

Armed with the knowledge presented in the previous sections, it is now possible to discuss the various ways in which IMS/VS interfaces with VSAM. The interface varies depending on the environment, online or batch (including IMS/VS utilities); the operating system; and finally the type of program that is being processed. Based on these combinations, a table is included as Figure A, Appendix A, which shows the VSAM buffer option which is selected by IMS/VS. The Buffer Code indicated is described in the Buffer Consideration Table, Figure C, Appendix A. The headings "DFSVSAMP REQUIRED" and "NATIVE VSAM USED" included in Figure C are analogous to LSR/GSR and NSR, respectively.

4.1 SELECTING A VALUE FOR BUFND

In summary, this parameter specifies the number of data buffers that are to be built for use by the designated data set if the NSR option is selected. If the data set is OPEN'ed to a shared resource pool (LSR/GSR), the specification will be ignored by VSAM OPEN. The parameter is coded in the control card of the data set as shown below:

```
//ABC    DD    DSN=.....,AMP='BUFND=n'
```

The value selected for BUFND affects the number of data buffers VSAM will use for read ahead operations when processing a data set sequentially. This value is also used by VSAM during data set creation in order to schedule multi-buffer data writes. A VSAM data set is considered in create mode when OPEN determines that the value of the data set high-used RBA (HURBA) is equal to zero. In some instances, IMS/VS OPEN's the data set, inserts a single record, and then issues a CLOSE. One of the functions performed by VSAM CLOSE is to update the catalog HURBA value of the data set being closed. At this time, the value would be incremented to greater than zero. The value of HURBA is always incremented and never decremented. The affect is such that a subsequent OPEN and a physical ERASE of the single record from the KSDS will not lower the value of HURBA. The data set will contain zero records, but will never again be in "create-mode". The only way to get back into create mode is to DELETE/DEFINE with Access Method Services (AMS) or to RESET the data set with the REUSE option.

The impact on buffering is that VSAM will schedule multi-buffer data writes when adding records to the data set only when in create mode. In non-create mode, a single data buffer, VSAM control interval, is written with each request to IOS.

Note this statement applies to adding records to a data set. When sequential operations for read-only or read for update are requested, VSAM performs chained reads and writes of data buffers.

A discussion of read-ahead buffering performed for sequential processing is included in the Palo Alto Systems Center Technical Bulletin, G320-6015, "OS/VS VSAM SHARING - A TECHNICAL DISCUSSION (PART I)". The discussion begins on page 65 of the bulletin and continues for several pages, covering both create and non-create mode processing. The following Figure 2 is an extract from a bulletin published by T. R. Mitchell and S. E. J. Friesenborg of the Washington Systems Center on VSAM performance. The figure illustrates the effect of buffer specification on elapsed time, as well as processing cycles and EXCP (EXCPVR or SVC121) requirements.

The results shown in Figure 2 indicate that the highest return is gained by overriding the default value up to a number that will allow scheduling full track I/O for each read or write operation. The effect of look-aside buffering can be seen on the read column as indicated by the increase in processing cycle requirements. The effect is such that the processing cycle requirements increase even though the number of EXCP's is decreasing.

The number of buffers VSAM schedules is based on the BUFND specification. It will schedule a minimum of the BUFND value divided by two. It then adds to that minimum value a percentage of the buffers remaining.

The percentage is calculated based on the number of records in a control interval. The algorithm states that, on an average, the cycles required to process a retrieved record is approximately 1/9 of the requirements to read a control interval from external storage. Therefore, the number of blocks of nine records in a control interval is computed by dividing control interval size by average record length and dividing the result by nine and rounding up. The resulting value minus one is then placed over the resulting value and this is the percentage of the remaining buffers which will be added to the originally calculated minimum. The purpose of this seemingly elaborate calculation is to provide optimum overlap of I/O with record processing. In the case of create mode processing, the end value is then rounded to a track boundary if the SPEED option was selected when the data set was defined with Access Method Services.

One final point is that VSAM will not schedule multiple buffers with a single I/O that will cross either a cylinder or a control area boundary. This could explain some cases with non- cylinder allocation where an increase in the BUFND value does not result in the anticipated reduction in

EXCP's.

FIGURE 2. EXAMPLE OF THE EFFECT OF BUFND SPECIFICATION ON SYSTEM PERFORMANCE

	READ			WRITE (CREATE MODE)		
BUFND	ELAPSED TIME	CPU TIME	EXCP COUNTS	ELAPSED TIME	CPU TIME	EXCP COUNTS
2	370.01	88.49	11713	222.79	69.30	10954
10	93.90	42.26	1703	127.22	67.35	2571
20	82.87	40.42	853	127.29	64.39	1720
30	83.71	40.72	640	122.38	62.70	1401
40	81.32	41.30	533	125.66	62.52	1295
50	81.41	41.65	427	138.93	61.31	1082
60	81.45	41.58	321	135.73	61.52	1082
70	82.92	42.72	321	139.80	62.33	1082
80	84.11	44.04	321	139.43	60.65	976
90	86.86	46.02	321	142.67	62.30	976
108	85.89	45.18	215	143.66	62.38	976

4.2 DFSVSAMP CONSIDERATIONS

The DFSVSAMP parameters are the way in which a user defines the amount and type of resources to be allocated to a pool. The requirements for a pool are generally dictated by an online, as opposed to batch, system. The intent of a pool is to maintain the resource in main storage to satisfy subsequent requests for the same block of data without incurring I/O to external storage. This is not the general characteristic of a batch program.

Characteristically, a batch program reads through an entire or a portion of a data set, each record sequentially. This is especially true of utilities such as HD and HS Unload and Reload. On the other hand, a utility such as DB Recovery with LOGS only input would operate more like an online environment relative to data base accessing patterns. The bulletin on OS/VS VSAM Sharing - A Technical Discussion (Part I) referenced earlier contains a section on the IBM DB/DC Systems and the IMS/VS section starts on page 33. This will provide the necessary assistance to aid the user in determining the resources to be allocated to the pool via the DFSVSAMP specifications.

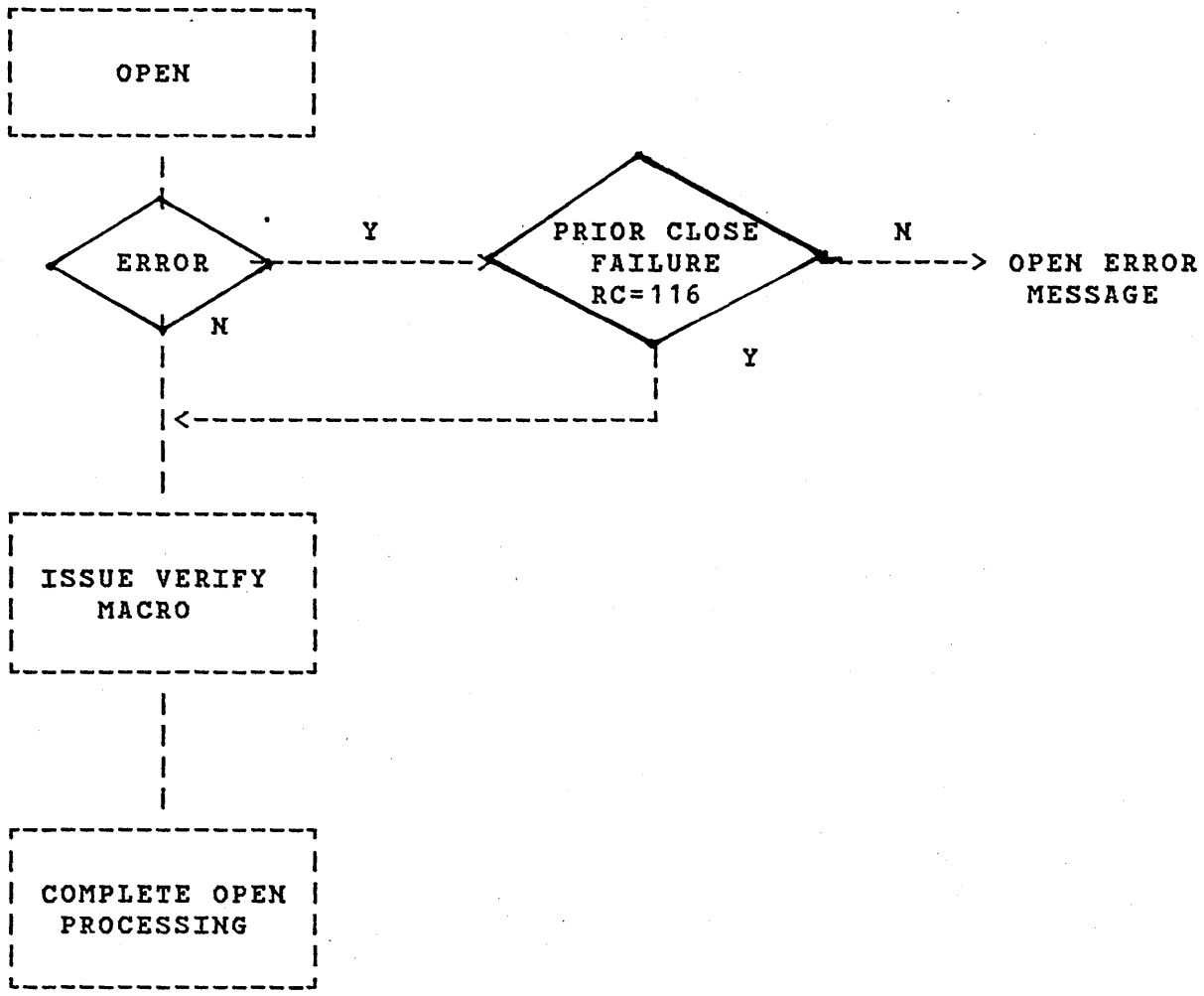
5.0 VERIFYING ON IMS/VS - VSAM DATA SET

The physical end of a VSAM data set is maintained by a logical, as opposed to hardware, indicator. This indication is noted by a control interval in which the last four bytes are binary zeroes. VSAM formatting operations always places this software end-of-file (SEOF) on a control area boundary in a key sequenced data set (KSDS), but may be on any control interval boundary within a control area of an entry sequenced data set (ESDS). The location of the SEOF is maintained in the VSAM catalog, user or master, into which the data set was defined. This is referred to as the High Used Relative Block Address (HURBA). This value is maintained for both the index and data components in a KSDS. The index component value is on any control interval boundary, as is the case with an ESDS.

There is a macro which is part of the VSAM system called VERIFY. This is the same macro which is issued internally by the Access Method Service VERIFY function. The purpose of VERIFY is to ensure that the catalog HURBA value(s) is not less than the formatted SEOF control interval. VERIFY validates the HURBA by accessing the data or index control interval pointed to by the corresponding current HURBA indicated in the catalog. This control interval is compared for a SEOF control interval (CIDF = binary zeroes). If it is not recognized as an SEOF control interval, the control intervals are accessed forward from that point until the SEOF is found or until the high allocated RBA (HARBA) is reached. If the SEOF is a lower RBA than the value of HURBA in the catalog, as could occur if an FDR restore is made of a down level version of the data set, "unpredictable results" will be encountered, i.e., I/O error, set HURBA=HARBA, etc.

The IMS/VS product internally issues the VERIFY macro. This is performed in batch utilities and user programs as well as in the online system whenever required. Beginning with IMS/VS 1.1.5, this is also performed by batch image copy. The VERIFY is performed even in cases where it is not absolutely necessary, but is done as a safeguard. Therefore, any IMS/VS program accessing a VSAM DL/I data base need not include an Access Method Services VERIFY jobstream immediately prior or subsequent to the execution of the IMS/VS program. The only case may be in SHISAM when the DL/I file is OPEN'ed as a native VSAM data set and accessed with non-DL/I requests. A flowchart is included as Figure 3 which outlines the logic implemented during VSAM data set OPEN by the IMS/VS system.

FIGURE 3. IMS/VS OPEN/VERIFY LOGIC



APPENDIX A - VSAM BUFFER OPTIONS TABLES

FIGURE A. IMS USE OF VSAM BUFFER OPTIONS

IMS REGION	OPERATION SYSTEM	PROGRAM TYPE	VSAM BUFFER OPTION	BUFFER CODE
ONLINE	VS2	ALL USER AND ONLINE IMAGE COPY	GSR*	B
			LSR	B
	VS1		LSR	B
BATCH	ALL OPERATING SYSTEMS	USER LOAD MODE: HISAM (KSDS ONLY) SHISAM	NSR (CREATE)	C
			LSR	B
		UTILITY LOAD MODE (HD RELOAD, HS RELOAD, - - - - - - - - - - DB RECOVERY)	LSR - - - - - - - - SEE FIGURE B	B - - -
		USER NON-LOAD MODE	LSR	B
		UTILITY NON-LOAD MODE (BACKOUT, HD UNLOAD, HS UNLOAD, - - - - - - - - - - IMAGE COPY)	LSR - - - - - - - - NSR (NON-CREATE)	B - - - D

* SINGLE IMS SYSTEM PER CPU

NOTE: Buffer Code entries are code values in the Buffer Consideration Table.

FIGURE B. DATA BASE RECOVERY (DFSURDB0)

INPUT	HISAM KSDS	HISAM ESDS	HIDAM PRIME INDEX KSDS	HIDAM ESDS	HDAM ESDS	SHISAM KSDS	SEC. INDEX KSDS	SEC. INDEX ESDS
IMAGE COPY	C	C	C	C	C	C	C	C
HISAM UNLOAD	C	A	C	N/A	N/A	C	C	A
C.A. WITHOUT DUMP	B	B	B	B	B	B	B	B
LOGS	B	B	B	B	B	B	B	B

NOTE: Individual entries are code values in the Buffer Consideration Table.

FIGURE C. BUFFER CONSIDERATION TABLE

CODE	DFSVSAMP REQUIRED	NATIVE VSAM USED	MODE*	BUFND RECOMMENDATIONS
A	Y	Y	U	MINIMUM = DEFAULT = STRNO + 1
B	Y	N	U	NOT APPLICABLE
C	N	Y	L	NUMBER OF CI'S PER TRACKS + 2
D	N	Y	U	2 TIMES THE NUMBER OF CI'S PER TRACK

* U = NON-CREATE
L = LOAD/CREATE

APPENDIX A - VSAM BUFFER OPTIONS TABLES

PALO ALTO SYSTEMS CENTER
READER'S COMMENT FORM

IMS/VS and OS/VS VSAM Buffer Options G320-6035
(Shared versus Non-Shared Resources)
Edward Daray, Dept. 65J

Please comment on the usefulness and readability of the publication, suggest additions and deletions, and list specific omissions and errors (give page numbers). IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

COMMENTS

Thank you for your cooperation. Fold this form on the two lines, tape, and mail to the address shown on the reverse side.

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Palo Alto Systems Center
1501 California Avenue
Palo Alto, California 94304

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601