

DB2® Server for VM



Data Spaces Support for VM/ESA

Version 6 Release 1

DB2® Server for VM



Data Spaces Support for VM/ESA

Version 6 Release 1

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page v.

This book is also provided as an online book that can be viewed with the IBM® BookManager® READ and IBM Library Reader™ licensed programs.

First Edition (December 1998)

This edition, SC09-2675, applies to Version 6 Release 1, of the IBM DATABASE 2™ Server for VSE & VM Program 5648-A70, and to all subsequent releases of this product until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or deletion.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to: IBM Canada Ltd. Laboratory, Information Development, 2G/345/1150/TOR, 1150 Eglinton Ave East, North York Ontario Canada M3C 1H7.

You can also send your comments by facsimile to (416) 448-6161 addressed to the attention of the RCF Coordinator. If you have access to Internet, you can send your comments electronically to **torrcf@ca.ibm.com**; IBMLINK™, to **toribm(torrcf)**; IBM/PROFS®, to **torolab4(torrcf)**; IBMMAIL, to **ibmmail(caibmw9)**; or through our home page at **http://www.software.ibm.com/data/db2/vse-vm**

If you choose to respond through Internet, please include either your entire Internet network address, or a postal address.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© **Copyright International Business Machines Corporation 1992, 1998. All rights reserved.**

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	v	VSE Guest Sharing	17
Trademarks	v	Enabling Requirements	17
About This Manual	vii	Operating System Overview	17
Who Should Use This Manual	vii	Virtual Machine Overview	17
Organization	vii	Software Requirements	18
Prerequisite Reading	viii	Virtual Storage Requirements	18
Syntax Notation Conventions	viii	Real Storage Requirements	19
Conventions Used for Highlighting Examples	xi	DASD Storage Requirements	19
How to Use EXECs	xii	Hardware Requirements	21
SQL Reserved Words	xii	Before Enabling	21
		Program Directory for DB2 Server for VM	21
		Preventive Service Planning	21
		Corrective Service	21
		Enabling Options	21
Summary of Changes for DB2 Version 6		Chapter 2. Enabling	23
Release 1	xv	Pre-Enable Checklist	23
Enhancements, New Functions, and New		Operating System Options	23
Capabilities	xv	Enabling for XC mode	23
DRDA RUOW Application Requestor for		Enabling for non-XC mode	23
VSE (Online)	xv	Enable Checklist	23
Stored Procedures	xv	Backing Up, Configuring and Enabling Your	
TCP/IP Support for DB2 Server for VM	xvi	Database Machine	25
New Code Page and Euro Symbol Code		Step 1: Log onto the MAINT Machine	25
Page Support	xvi	Step 2: Update the CP Directory	25
DataPropagator™ Capture	xvi	Step 3: Log off the MAINT Machine	26
QMF for VM, QMF for VSE, and QMF for		Step 4: Log onto the SQLMACH Machine	27
Windows®	xvii	Step 5: Archive your Database	27
RDS Above the Line	xvii	Step 6: Activate VMDSS	27
Combining of NLS Feature Installation		Step 7: Log off the SQLMACH Machine	28
Tapes with Base Product Installation		Step 8: Log onto the DB2 for VM	
Tape	xvii	Installation User ID (564815A1)	28
Control Center Feature	xviii	Step 9: Link-Edit the Load Library	28
Data Restore Feature	xviii	Step 10: Resave the DBSS Saved Segment	29
DB2 REXX SQL Feature	xviii	Step 11: Stop the Application Server	32
Reliability, Availability, and Serviceability		Step 12: Add New Users to the Database	32
Improvements	xviii	Step 13: Change the Database	
Migration Considerations	xviii	Administrator Password	32
Library Enhancements	xix	Step 13A: Verify non-XC Mode Installation	33
		Step 13B: Verify XC Mode Installation	34
		Step 14: System Customization Activities	36
		Disabling VMDSS	36
		Disable Step 1: Archive your Database	36
		Disable Step 2: Access the Service Disk or	
		Directory	36
		Disable Step 3: Reblock the Directory Disk	36
		Disable Step 4: Remove the VMDSS Files	36
		Disable Step 5: Link-Edit the Load Library	37
		Disable Step 6: Restart the Application	
		Server	37
Chapter 1. Before You Begin	1		
Improving DB2 Server for VM Performance	1		
Understanding VM Data Spaces	1		
Understanding how VMDSS uses Data			
Spaces	4		
Storage Pools	8		
Internal Dbspaces	9		
Directory	10		
Managing Main and Expanded Storage	11		
Striping	13		
Performance Counters	14		
Planning Structure by Storage Pool	14		
Logical and Physical Mapping	15		

Chapter 3. Operating	39	Choosing Storage Residence Priorities . . .	73
Storage Pool Specifications	39	Unmapped Internal Dbspaces	74
Changing Storage Pool Specifications at Startup	39	Managing Checkpoints	75
Checking Your Current Storage Pool Specifications	41	Choosing the Checkpoint Interval	75
Changing Storage Pool Specifications Dynamically	41	Choosing the Save Interval	76
Application Server Initialization Parameters	42	Using Striping	76
Changing Initialization Parameters at Startup	42	77
Displaying TARGETWS and SAVEINTV	43	Choosing Logical or Physical Mapping	77
Changing TARGETWS and SAVEINTV Dynamically	43	Real Storage Requirements for Data Spaces	77
Using Data Spaces with Internal Dbspaces	44	Appendix A. EXECs	79
Unmapped Internal Dbspaces	44	SQLCDBEX	79
Mapped Internal Dbspaces	44	Syntax	79
Using Data Spaces with the Directory	45	SQLDBGEN	80
Reblocking the Database Directory	45	Syntax	80
Using Data Spaces Support with a New Database	48	SQLDBINS	82
Using Storage Pool Performance Counters	48	Syntax	83
Displaying Pool Counters	49	Appendix B. Storage Pool Specification	
Resetting Pool Counters	50	File Format	85
Checking the Status of Users	50	File Format	85
Looking at the Buffer Pools	52	Data Line Syntax	85
Chapter 4. Measuring Performance	55	Ordering Data Lines	86
Understanding Performance Measurements	55	Specification File Example	87
Relative Measurements	55	Appendix C. DB2 Server for VM	
Sampling Interval	56	Initialization Parameters	89
Measurement Tools	56	Example	89
CP Monitor and VM/PRF	57	Appendix D. Determining Number of Data Spaces	91
CP INDICATE USER and QUERY TIME Commands	57	Maximum Number of Data Spaces	91
DB2 Server for VM COUNTER Operator Command	59	Logical Mapping	91
VMDSS COUNTER POOL Operator Command	61	Physical Mapping	93
Chapter 5. Tuning Performance	69	Maximum Total Size	96
Deciding When to Use Data Spaces	69	Displaying Current Data Spaces	96
Advantages	69	Appendix E. Internal Counters	97
Storage Pool	71	Appendix F. Why is the TARGETWS Value Frequently Exceeded?	101
Internal Dbspaces	71	VMDSS Usage Scenario	102
Directory	72	Bibliography	105
Managing Your Working Storage Size	72	Index	109
Choosing the Target Working Storage Size	73		

Notices

Any references to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, U.S.A.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Canada Ltd., Department 071, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7, Canada. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States and/or other countries:

IBM
BookManager
CICS/VSE
DataPropagator
DATABASE 2
DB2
ES/9000
IMS
Library Reader
OS/390
QMF
SQL/DS
VM/ESA
VSE/ESA

This publication could contain technical inaccuracies or typographical errors.

Lotus and Lotus Notes are trademarks of Lotus Development Corporation in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of others.

About This Manual

Who Should Use This Manual

This manual is for the person who will operate, tailor, or tune the VM Data Spaces Support on a VM/ESA® operating system.

Organization

Chapter 1, “Before You Begin” on page 1 presents:

- An overview of the VM Data Spaces Support
- The hardware and software you need to enable it
- The information you need to plan to enable, operate, and tune it.

Chapter 2, “Enabling” on page 23 presents the enabling steps.

Chapter 3, “Operating” on page 39 describes operating tasks.

Chapter 4, “Measuring Performance” on page 55 describes performance measurements and how to interpret them.

Chapter 5, “Tuning Performance” on page 69 describes the settings and options that are provided for tuning.

Appendix A, “EXECs” on page 79 lists and describes the updated CMS EXECs included with this product.

Appendix B, “Storage Pool Specification File Format” on page 85 describes the format and syntax of the control file used to tailor VMDSS.

Appendix C, “DB2 Server for VM Initialization Parameters” on page 89 describes the additional DB2 Server for VM initialization parameters you can use.

Appendix D, “Determining Number of Data Spaces” on page 91 describes how to calculate the maximum number of data spaces you will use with your database and their total size.

Appendix E, “Internal Counters” on page 97 describes a set of counters that you can use to monitor the internal operation of VMDSS.

Appendix F, “Why is the TARGETWS Value Frequently Exceeded?” on page 101 describes how TARGETWS operates and how VM/ESA manages real storage.

Prerequisite Reading

This manual assumes that you are familiar with the following IBM publications:

- *DB2 Server for VM Program Directory*
- *DB2 Server for VM Messages and Codes*
- *DB2 Server for VM System Administration*
- *DB2 Server for VSE & VM Operation.*

It also assumes you are familiar with VM/ESA systems, CMS commands, and EXECs.

Syntax Notation Conventions

Throughout this manual, syntax is described using the structure defined below.

- Read the syntax diagrams from left to right and from top to bottom, following the path of the line.

The >>— symbol indicates the beginning of a statement or command.

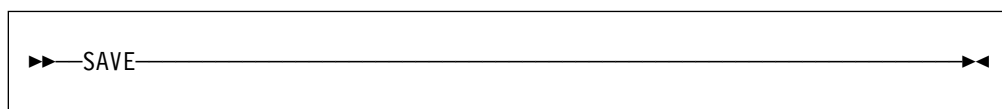
The —> symbol indicates that the statement syntax is continued on the next line.

The >— symbol indicates that a statement is continued from the previous line.

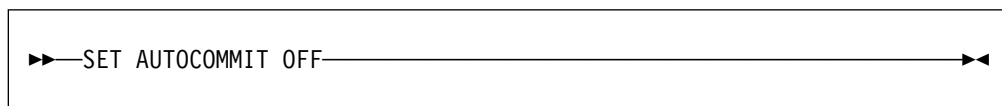
The —>< symbol indicates the end of a statement.

Diagrams of syntactical units that are not complete statements start with the >— symbol and end with the —> symbol.

- Some SQL statements, Interactive SQL (ISQL) commands, or database services utility (DBS Utility) commands can stand alone. For example:



Others must be followed by one or more keywords or variables. For example:



- Keywords may have parameters associated with them which represent user-supplied names or values. These names or values can be specified as either constants or as user-defined variables called *host_variables* (*host_variables* can only be used in programs).
- Keywords appear in either uppercase (for example, `SAVE`) or mixed case (for example, `CHARacter`). All uppercase characters in keywords must be present; you can omit those in lowercase.
- Parameters appear in lowercase and in italics (for example, *synonym*).
- If such symbols as punctuation marks, parentheses, or arithmetic operators

►► DROP SYNONYM *synonym* ◀◀

are shown, you must use them as indicated by the syntax diagram.

- All items (parameters and keywords) must be separated by one or more blanks.
- Required items appear on the same horizontal line (the main path). For example, the parameter *integer* is a required item in the following command:

►► SHOW DBSPACE *integer* ◀◀

This command might appear as:

```
SHOW DBSPACE 1
```

- Optional items appear below the main path. For example:

►► CREATE
└─ UNIQUE ─┘ INDEX ◀◀

This statement could appear as either:

```
CREATE INDEX
```

or

```
CREATE UNIQUE INDEX
```

- If you can choose from two or more items, they appear vertically in a stack.

If you must choose one of the items, one item appears on the main path. For example:

►► SHOW LOCK DBSPACE
└─ ALL ─┘
└─ *integer* ─┘ ◀◀

Here, the command could be either:

```
SHOW LOCK DBSPACE ALL
```

or

```
SHOW LOCK DBSPACE 1
```

If choosing one of the items is optional, the entire stack appears below the main path. For example:

►► BACKWARD
└─ *integer* ─┘
└─ MAX ─┘ ◀◀

Here, the command could be:

BACKWARD

or

BACKWARD 2

or

BACKWARD MAX

- The repeat symbol indicates that an item can be repeated. For example:



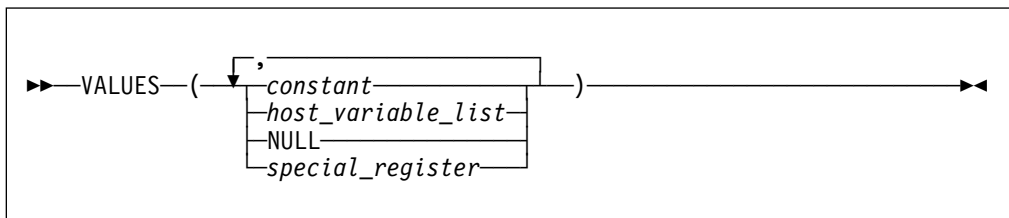
This statement could appear as:

ERASE NAME1

or

ERASE NAME1 NAME2

A repeat symbol above a stack indicates that you can make more than one choice from the stacked items, or repeat a choice. For example:



- If an item is above the main line, it represents a default, which means that it will be used if no other item is specified. In the following example, the ASC keyword appears above the line in a stack with DESC. If neither of these values is specified, the command would be processed with option ASC.

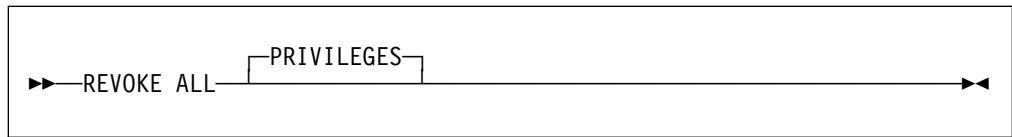


- When an optional keyword is followed on the same path by an optional default parameter, the default parameter is assumed if the keyword is not entered. However, if this keyword is entered, one of its associated optional parameters must also be specified.

elln the following example, if you enter the optional keyword PCTFREE =, you also have to specify one of its associated optional parameters. If you do not enter PCTFREE =, the database manager will set it to the default value of 10.

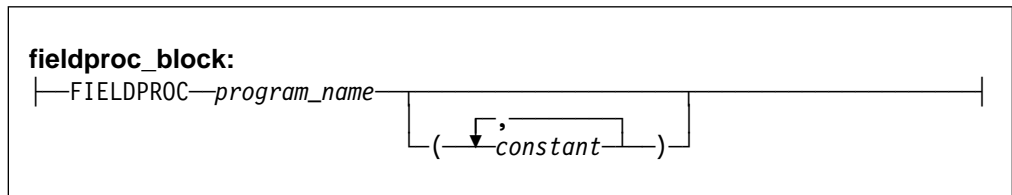
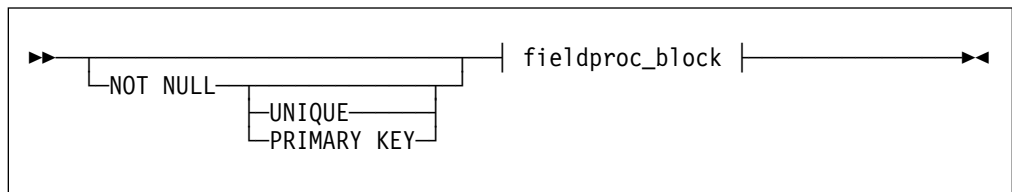


- Words that are only used for readability and have no effect on the execution of the statement are shown as a single uppercase default. For example:



Here, specifying either REVOKE ALL or REVOKE ALL PRIVILEGES means the same thing.

- Sometimes a single parameter represents a fragment of syntax that is expanded below. In the following example, **fieldproc_block** is such a fragment and it is expanded following the syntax diagram containing it.



Conventions Used for Highlighting Examples

Sample commands and messages are provided throughout this manual. While you will not see highlighting on your screen, it is included in this manual for emphasis:

- Commands are **highlighted using bold type**.
- Messages are not highlighted
- Important parts of some messages are emphasized with underlining.

For example:

```

set pool 1 seq
ARI0065I Operator command processing is complete.
show pool 1

POOL NO. 1:      NUMBER OF EXTENTS = 2  DS3 SEQ

EXTENT  TOTAL  NO. OF  NO. OF  NO. OF  %
NO.     PAGES  PAGES  USED   FREE   PAGES  RESV  PAGES  USED
  1      855    74     781                    8
  2      855    47     808                    5
TOTAL   1710   121    1589          20    7
ARI0065I Operator command processing is complete.

```

How to Use EXECs

When running the EXECs included with DB2 Server for VM, you may receive a message like the following:

```

Do you want to use these values?
Enter 0 (No) or 1 (Yes) or 111 (Quit)

```

To answer no, enter one of the following:

- NO, No, no
- N, n
- 0

To answer yes, enter one of the following:

- YES, Yes, yes
- Y, y
- 1

To stop an EXEC and exit processing, enter one of the following:

- QUIT, Quit, quit
- 111

SQL Reserved Words

The following words are reserved in the SQL language. They cannot be used in SQL statements except for their defined meaning in the SQL syntax or as host variables, preceded by a colon.

In particular, they cannot be used as names for tables, indexes, columns, views, or dbspaces unless they are enclosed in double quotation marks ("").

ACQUIRE	GRANT	RESOURCE
ADD	GRAPHIC	REVOKE
ALL	GROUP	ROLLBACK
ALTER		ROW
AND	HAVING	RUN
ANY		
AS	IDENTIFIED	SCHEDULE
ASC	IN	SELECT
AVG	INDEX	SET
	INSERT	SHARE
BETWEEN	INTO	SOME
BY	IS	STATISTICS
		STORPOOL
CHAR	LIKE	SUM
CHARACTER	LOCK	SYNONYM
COLUMN	LONG	
COMMENT		TABLE
COMMIT	MAX	TO
CONCAT	MIN	
CONNECT	MODE	UNION
COUNT		UNIQUE
CREATE	NAMED	UPDATE
CURRENT	NHEADER	USER
	NOT	
DBA	NULL	VALUES
DBSPACE		VIEW
DELETE	OF	
DESC	ON	WHERE
DISTINCT	OPTION	WITH
DOUBLE	OR	WORK
DROP	ORDER	
EXCLUSIVE	PACKAGE	
EXECUTE	PAGE	
EXISTS	PAGES	
EXPLAIN	PCTFREE	
	PCTINDEX	
FIELDPROC	PRIVATE	
FOR	PRIVILEGES	
FROM	PROGRAM	
	PUBLIC	

Summary of Changes for DB2 Version 6 Release 1

This is a summary of the technical changes to the DB2 Server for VSE & VM Version 6 Release 1 database management system. All manuals are affected by some or all of the changes discussed here. This summary does not list incompatibilities between releases of the DB2 Server for VSE & VM product; see either the *DB2 Server for VSE & VM SQL Reference*, *DB2 Server for VM System Administration*, or the *DB2 Server for VSE System Administration* manuals for a discussion of incompatibilities. Version 6 Release 1 of the DB2 Server for VSE & VM database management system is intended to run on the Virtual Machine/Enterprise Systems Architecture (VM/ESA®) Version 2 Release 2 or later environment and on the Virtual Storage Extended/Enterprise Systems Architecture (VSE/ESA™) Version 2 Release 2 or later environment.

Enhancements, New Functions, and New Capabilities

DRDA RUOW Application Requestor for VSE (Online)

DRDA Remote Unit of Work Application Requestor provides read and update capability in one location in a single unit of work.

This support provides CICS/VSE® online application programs with the ability to execute SQL statements to access and manipulate data managed by any remote application server that implements the DRDA architecture. Online application programs that access remote application servers need to be preprocessed to create a bind file and then bound (using CBND) to the remote application server. Online application programs that access a local application server are preprocessed as in previous releases.

See the following DB2 Server for VSE & VM manuals for further information:

- *DB2 Server for VSE System Administration*
- *DB2 Server for VSE & VM SQL Reference*
- *DB2 Server for VSE Database Administration*
- *DB2 Server for VSE Application Programming*
- *DB2 Server for VSE Installation*

Stored Procedures

The ability to use stored procedures provides distributed solutions that let more people access data faster.

A stored procedure is a user-written application program compiled and stored at the server. When the database is running in multiple user mode, local applications or remote DRDA applications can invoke the stored procedure. SQL statements are local to the server and issued by a stored procedure so they do not incur the high network costs of distributed statements. Instead, a single network send and receive operation is used to invoke a series of SQL statements contained in a stored procedure.

See the following DB2 Server for VSE & VM manuals for further information:

- *DB2 Server for VM System Administration*
- *DB2 Server for VM Database Administration*
- *DB2 Server for VSE & VM SQL Reference*
- *DB2 Server for VSE & VM Operation*

TCP/IP Support for DB2 Server for VM

TCP/IP support allows:

- VM applications to use SQLDS-private protocol to connect to VM databases over TCP/IP.
- VM applications to use DRDA protocol to connect to DB2 family databases (and any other database that supports DRDA connections) over TCP/IP.
- non-VM applications to use DRDA-protocol to access VM database over TCP/IP.

TCP/IP support for DB2 Server for VM integrated with the DB2 Server for VM application server means a system easier to configure and maintain.

The database manager will optionally secure TCP/IP connections using any external security manager that supports the RACROUTE interface.

New Code Page and Euro Symbol Code Page Support

The following CCSIDs are now supported:

- 1112: Latvian/Lithuanian
- 1122: Estonian
- 1123: Ukrainian
- 1130: Vietnamese
- 1132: Lao
- 1148: E-International
- 1140: E-English
- 1141: E-German
- 1144: E-Italian
- 1147: E-French

Additional support has been added for conversions from Unicode (UCS-2) to host CCSIDs.

For a complete list of CCSIDs supported refer to the *DB2 Server for VM System Administration* and *DB2 Server for VSE System Administration* manuals.

DataPropagator™ Capture

DataPropagator Capture is part of the DB2 Family of DataPropagator products. DataPropagator Capture is updated for Version 6 Release 1 compatibility.

QMF for VM, QMF for VSE, and QMF for Windows®

IBM Query Management Facility (QMF™) is now an separately priced feature of DB2 Server for VSE & VM. QMF is a tightly integrated, powerful, and reliable tool that performs query and reporting for IBM's DB2 relational database Management System Family. It offers an easy-to-learn, interactive interface. Users with little or no data processing experience can easily retrieve, create, update, insert, or delete data that is stored in DB2.

QMF offers a total solution that includes accessing large amounts of data and sharing central repositories of queries and enterprise reports. It also allows you to implement tightly-controlled, distributed, or client-server solutions. In addition, you can use QMF to publish reports to the World Wide Web that you can view with your favorite web browser.

Using QMF, users can access a wide variety of data sources, including operational or warehouse data from many platforms: DB2 for VSE, VM, OS/390® and Windows. Via IBM Data Joiner, you can access non-relational data, such as IMS™ and VSAM, as well as data from other vendor platforms.

RDS Above the Line

The RDS component will load and execute above the 16 megabyte line. This support frees up approximately 1.5 megabytes of storage below the 16 megabyte line (or approximately 2.5 megabytes, if DRDA is installed) when compared to Version 5 Release 1. No installation or migration changes are required for this support to be used (except for the definition of VM Shared Segments and for users who execute the database server with AMODE(24)). If sufficient storage is available, the RDS component will be automatically loaded above the 16 megabyte line. When using VM Shared Segments, the RDS Segment should be defined above the 16 megabyte line.

VM users who wish to run the database server in 24-bit addressing mode (i.e. use the AMODE(24) parameter) **must** use a virtual storage size no greater than 16 megabytes. See the *DB2 Server for VM System Administration* or *DB2 Server for VSE System Administration* for release to release incompatibility information.

Combining of NLS Feature Installation Tapes with Base Product Installation Tape

All available NLS features for DB2 Server for VSE, DB2 Server for VM, Control Center for VSE and REXX SQL for VM have been combined with the respective base product installation tape. Customers interested in an NLS feature language will no longer need to order an additional NLS feature tape because all NLS languages will be available to all customers. In all cases, the default language as shipped is American English. The installation and migration processes have been changed to allow you to choose the default language. Refer to the *DB2 Server for VM Program Directory*, *DB2 Server for VSE Installation*, *DB2 for VSE Control Center Installation and Operations Guide*, and *DB2 REXX SQL for VM/ESA Installation* for the details of how these changes affect the installation process and how you can choose to have a different default language.

Control Center Feature

DB2 Server for VSE & VM Version 6 Release 1 enhances the new Control Center feature as follows:

For both VM/ESA and VSE/ESA:

- Access to the Query Management Facility (QMF)

For VM/ESA:

- Compatibility with DB2 Server for VM Version 6 Release 1 initialization parameters and operator commands
- Shared File System Support (SFS) in a VM/ESA environment
- CA-DYNAM/T Interface Support in a VM/ESA environment
- Data Restore Incremental Backup Support in a VM/ESA environment

For VSE/ESA:

- Control Center code installation on any library
- Ability to use while viewing a list of tables online
- Ability to create, reorganize, unload, reload, move and copy tables in batch mode
- Ability to update table statistics in batch mode
- Ability to drop tables online

Data Restore Feature

The Data Restore feature provides archiving and recovery functions in addition to those provided in DB2 for VSE & VM. Data Restore is enhanced in Version 6 Release 1 with incremental database archiving support. The support allows you to archive only the areas of the database that have been updated since the last database archive, instead of having to archive the entire database. This can provide significant savings for customers with large databases which are updated infrequently, or where only a small fraction of the database is updated frequently.

DB2 REXX SQL Feature

The DB2 REXX SQL feature provides a REXX interface for VM customers to allow SQL calls to be executed from REXX programs. The DB2 REXX SQL feature is updated for Version 6 Release 1 compatibility.

Reliability, Availability, and Serviceability Improvements

Migration Considerations

Migration is supported from SQL/DS™ Version 3 and DB2 Server for VSE & VM Version 5. Migration from SQL/DS Version 2 Release 2 or earlier releases is not supported. Refer to the *DB2 Server for VM System Administration* or *DB2 Server for VSE System Administration* manual for migration considerations.

Library Enhancements

Some general library enhancements include:

- The following books have been removed from the library:
 - *DB2 Server for VM Operation*
 - *DB2 Server for VSE Operation*
 - *DB2 Server for VM Interactive SQL Guide and Reference*
 - *DB2 Server for VSE Interactive SQL Guide and Reference*
 - *DB2 Server for VM Database Services Utility*
 - *DB2 Server for VSE Database Services Utility*
- The following books have been added to the library:
 - *DB2 Server for VSE & VM Operation*
 - *DB2 Server for VSE & VM Interactive SQL Guide and Reference*
 - *DB2 Server for VSE & VM Database Services Utility*

Refer to the new *DB2 Server for VSE & VM Overview* for a better understanding of the benefits DB2 Server for VSE & VM can provide.

Chapter 1. Before You Begin

Read this chapter before you implement VM Data Spaces Support (VMDSS). It briefly describes the concept of data spaces, how they work, and how they can improve performance; it outlines what options you have as a VMDSS user; and it lists the prerequisite hardware and software.

Improving DB2 Server for VM Performance

VMDSS can dramatically increase the performance of your application server, by using the Data Spaces facility found in VM/ESA. Data spaces give your application server access to vast amounts of fast storage, and uses a high performance DASD I/O system that has many advantages over the standard I/O system (IUCV *BLOCKIO).

VMDSS can also distribute data across multiple dbextents, which helps to balance the load on your system's DASD and allows the operating system to read and write data in parallel. Finally, you can monitor the performance of your DASD I/O system for individual storage pools, and control the amount of system-resource it uses.

Understanding VM Data Spaces

To understand how data spaces work and why they are an improvement over existing systems, you first need to understand how VM uses its *paging system* to manage virtual machine storage.

Standard Virtual Machine Storage

Each virtual machine within a VM system has its own *virtual address space* (also called a primary address space) which is where you load and run programs. Because this space is *virtual*, the operating system does not dedicate a piece of main storage (also called real storage) to each virtual machine. You do not need to buy 16MB of main storage for each 16MB virtual machine. Instead the operating system only uses main storage for those parts of virtual storage you need right now, or are likely to need in the near future.

These parts of storage are divided into 4KB (4096 byte) blocks called *pages*. When a virtual machine needs a page that it has not accessed before, the operating system retrieves it from its location on DASD, and loads it into an empty page in main storage. (Before a virtual machine can use a page, it must be in main storage.)

When the operating system runs out of free pages in main storage, it moves the least recently used ("oldest") page to auxiliary storage to create a free space for a new page. While the virtual machine is still active (logged on), the page will remain in either main or auxiliary storage.

VM/ESA uses two types of auxiliary storage: system paging DASD, and optional expanded storage. If your system has expanded storage, a page will be moved there first. If expanded storage is full, the least recently used page in expanded storage is moved to system paging DASD by way of main storage. When a virtual machine needs a page that it has previously used, the operating system moves it back to main storage from expanded storage, or from system paging DASD, if it is not already in main storage.

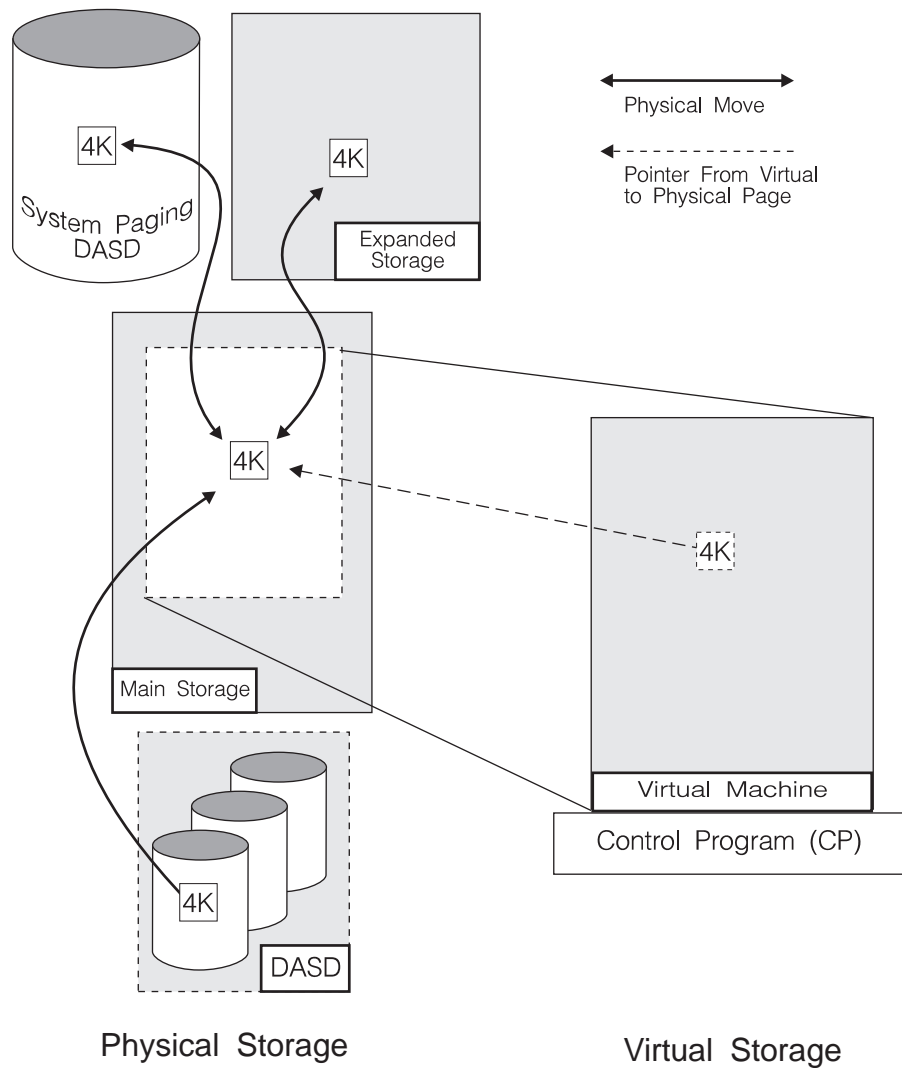


Figure 1. Standard Virtual Machine Storage

This paging system accomplishes two things. First, it allows each virtual machine to use much more storage than could be accommodated in main storage alone. Second, it keeps the most recently used pages in the storage devices that are the fastest to access. (The most recently used pages are the ones most likely to be used again in the near future.) Main and expanded storage are much faster than system paging DASD, and while expanded storage can be as fast as main storage, it is effectively slower because the operating system still needs to move the page into main storage before it can use it.

Data Spaces Storage

In VM/ESA, a program running in a machine's primary space can dynamically create additional address spaces for data, called *data spaces*. Like a virtual machine's primary address space, a data space is a virtual space with its real pages in main storage, in expanded storage, and on DASD. However, unlike a primary space, you cannot run a program in a data space. Also, in VMDSS the VM paging system manages data space pages differently than virtual machine pages. This means that data spaces do not use system paging DASD. (Unmapped internal dbspaces are the exception. Refer to "Unmapped Internal Dbspaces" on page 9).

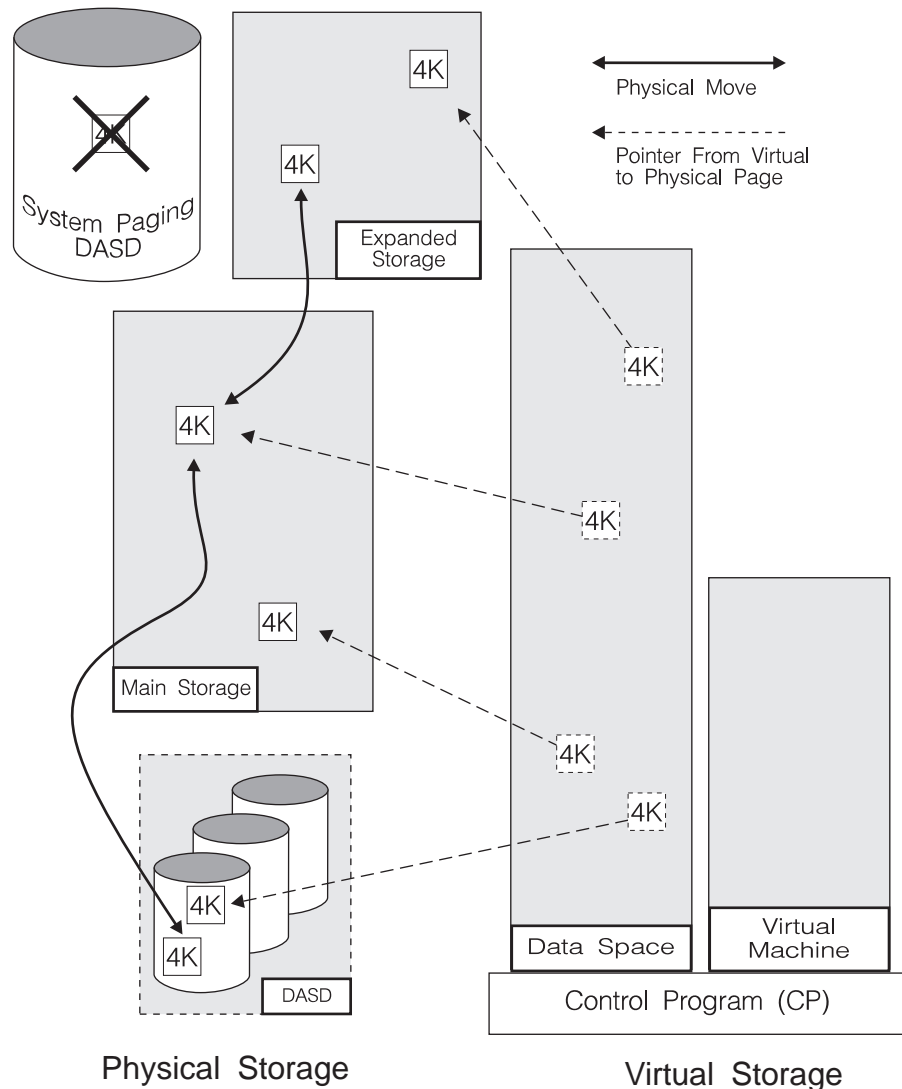


Figure 2. Data Spaces Storage

With VMDSS, if there is no longer any free space in main or expanded storage, the operating system will simply replace an old data space page in main or expanded storage with a new page. If the old page is needed again, it is reread from its

original DASD source. If the old page was modified while it was in main storage, the operating system ensures that the modified page is written back to its original DASD source before it is overwritten.

This expands a machine's effective virtual storage by providing additional addresses for data, thereby making room in the primary space for larger programs.

Understanding how VMDSS uses Data Spaces

Reading Pages

Before a page of data can be used by the database manager, it must be located in its data page *buffers*. The buffers are two areas of storage in your primary address space, which are created when you start the database manager. One area is reserved for pages from the directory disk, and the other for pages from storage pools. They are called the *directory buffer pool* and the *local buffer pool*, respectively.

With Data Spaces Support off: When the database manager needs a page, it looks for it in its buffer pool. If it does not find it there, it uses a VM service called IUCV *BLOCKIO to read the page from DASD into a free space in its pool.

Since the buffer pools are part of a primary address space, the operating system treats them like part of the database manager code. If a buffer page is not referenced frequently, it may be moved out to expanded storage or system paging DASD by the VM paging system (for more information refer to "Asynchronous Page Fault Processing" on page 69).

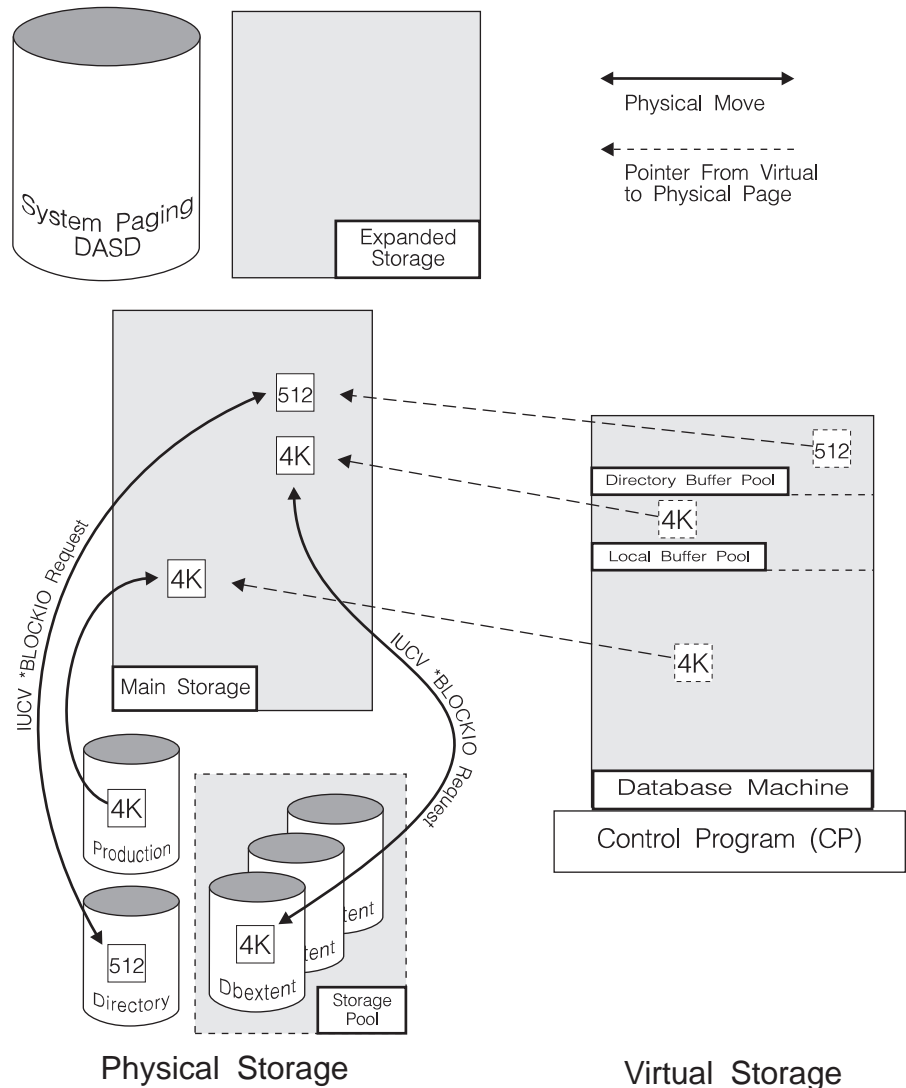


Figure 3. Page Movement in the Standard DASD I/O System. The database manager explicitly directs the operating system to move pages to and from DASD with the IUCV *BLOCKIO instruction. Once database machine pages are in main storage, they may be moved out to either expanded storage or system paging DASD by the VM paging system.

With Data Spaces Support on: If the database manager cannot find a page in its buffer pools, it “retrieves” it from a data space and stores it in a free buffer in its pool. When this happens, the operating system actually does most of the work. If the page is already in main storage, the database manager can move it directly to the buffer pool. If the page is in expanded storage or in DASD, the operating system moves it into main storage, and then copies it into a buffer. If your processor supports *Enhanced Move Page for VM*, pages are moved from expanded storage directly into a buffer.

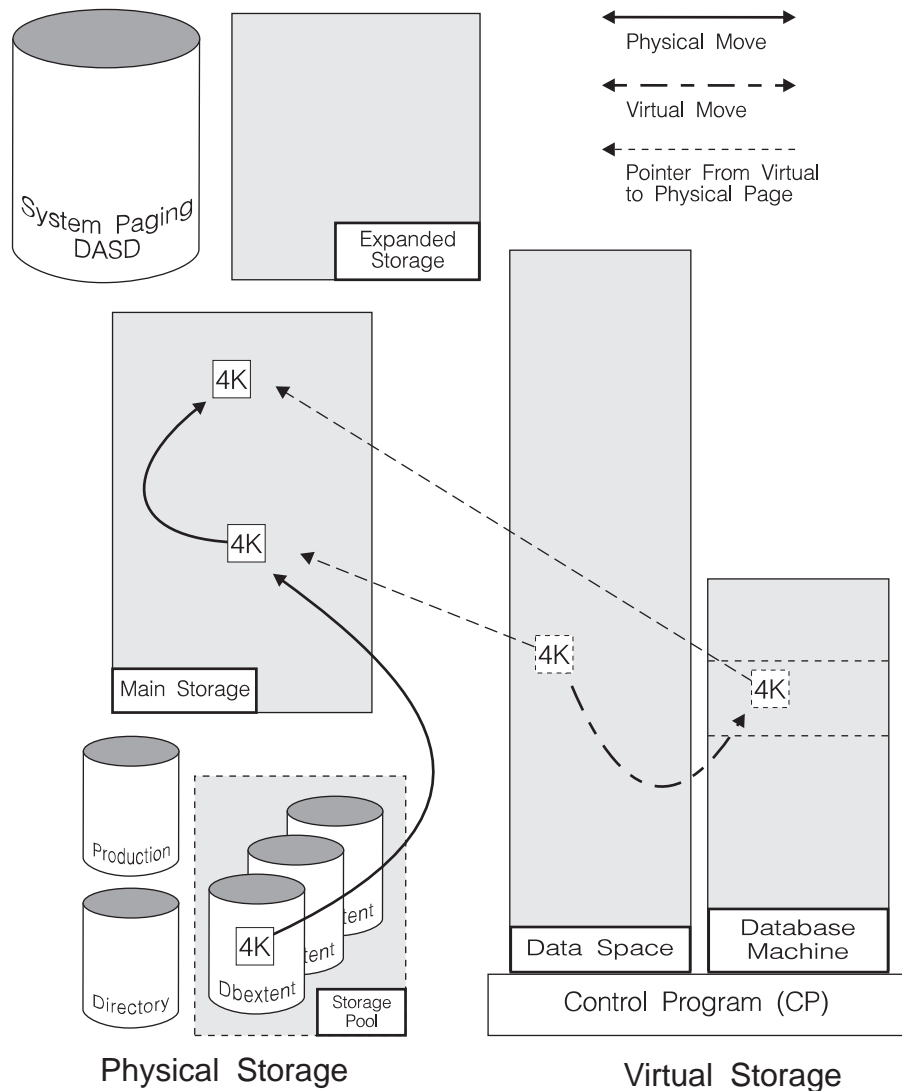


Figure 4. Page Movement with Data Spaces Support. The figure shows a page being read into main storage from DASD and then into a buffer pool. The page will be in two places in main storage until either the data space page is released or the database manager releases the buffer page (refer to "Releasing Pages" on page 7).

This system has several advantages over the standard DASD I/O system, which uses the IUCV *BLOCKIO service. In the latter, each page move must be explicitly requested by the database manager, whereas here paging is done by the VM paging system. This is faster and more efficient for several reasons, including:

- Shorter path length
- Asynchronous page fault processing
- Striping
- Blocking and prefetching
- Dynamic storage size management
- More asynchronous writes.

Refer to “Deciding When to Use Data Spaces” on page 69 for a detailed description of each advantage.

Releasing Pages

With Data Spaces Support off: When the database manager needs a buffer for another page, it overwrites the “oldest” unmodified page in the pool with a new page. This is referred to as *releasing* a page or *stealing* a buffer.

With Data Spaces Support on: When the database manager needs a buffer for another page, it does so in the same way that it would with Data Spaces Support off.

When the operating system needs main or expanded storage for itself or for other virtual machines, it may release data space pages from main storage. Pages may also be released at the request of the database manager. There are several parameters you can use to control when the database manager will start releasing pages and which pages it will choose (refer to “Managing Main and Expanded Storage” on page 11).

Modifying Pages

With Data Spaces Support off: While a page is in the buffer pool, the database manager may modify it. To ensure the integrity of your data, a modified page will not be released until it has been written back to DASD. If the database manager needs a buffer occupied by a modified page, it first writes the page to DASD, then loads the buffer with a new page.

With Data Spaces Support on: Instead of writing the modified page to DASD, the database manager moves it to a data space.

Once again, the operating system does most of the work. It takes the modified page from the database manager and moves it into main or expanded storage.

If the operating system needs a main storage page that is occupied by that page, it will move it to expanded storage or to DASD before it uses the main storage page. Similarly, if it needs an expanded storage page that is occupied by a modified page, it will move the modified page to DASD by way of main storage.

Checkpoints

At checkpoints, the database manager writes a summary status record to the log and makes sure that all modified buffer pages and status information are written to DASD. This ensures that you have a permanent record of your data on DASD.

With Data Spaces Support off: The database manager writes all the modified pages that are still in the buffer pools to DASD. Until it is finished it cannot serve any users.

With Data Spaces Support on: The database manager moves modified pages that are still in the buffer pools to data spaces. It then directs the operating system to save all the data space pages that were modified by the database manager to DASD. Until the operating system is finished, the database manager is forced to wait: it cannot serve any users until the checkpoint is complete.

Storage Pools

Individual storage pools can be used with or without Data Spaces Support.

Note: A storage pool used only for internal dbspaces and which has a dbextent on a virtual disk cannot be used with Data Spaces Support turned on for this pool. For more information on virtual disk support for VM/ESA for internal dbspaces, refer to *DB2 Server for VSE & VM Performance Tuning Handbook*.

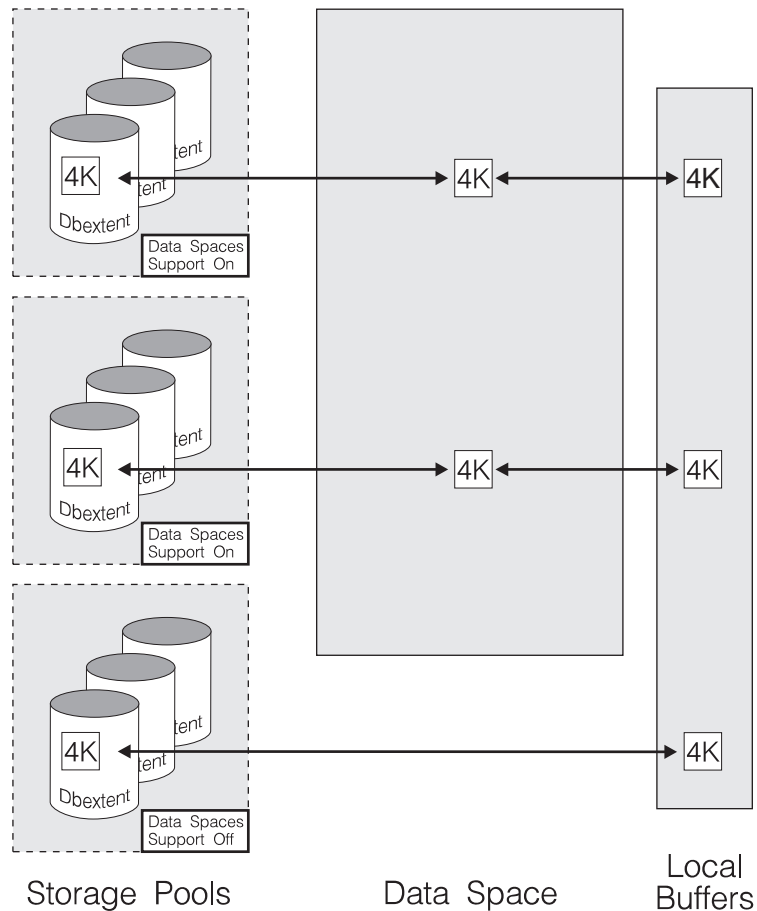


Figure 5. Storage Pools with Data Spaces Support on and off

For a description of how to turn on Data Spaces Support, refer to “Specifying Either Data Spaces Support or Standard DASD I/O” on page 40. For information on when to use data spaces with storage pools, refer to “Storage Pool” on page 71.

Internal Dbspaces

Internal dbspaces can be used with or without Data Spaces Support.

Mapped Internal Dbspaces

Since internal dbspaces are assigned to a specific storage pool, you can turn Data Spaces Support on and off for them by turning support on and off for that pool. This type of Data Spaces Support is similar to the support for any other storage pool. Since the system assigns, or “maps” each data space page to a physical page in a dbextent (contained in a storage pool), it is referred to as *mapped* support.

Unmapped Internal Dbspaces

VMDSO also supports *unmapped* internal dbspaces. Instead of mapping pages onto dbextents, VM/ESA manages them as normal virtual storage paged to VM system paging DASD (see Figure 6).

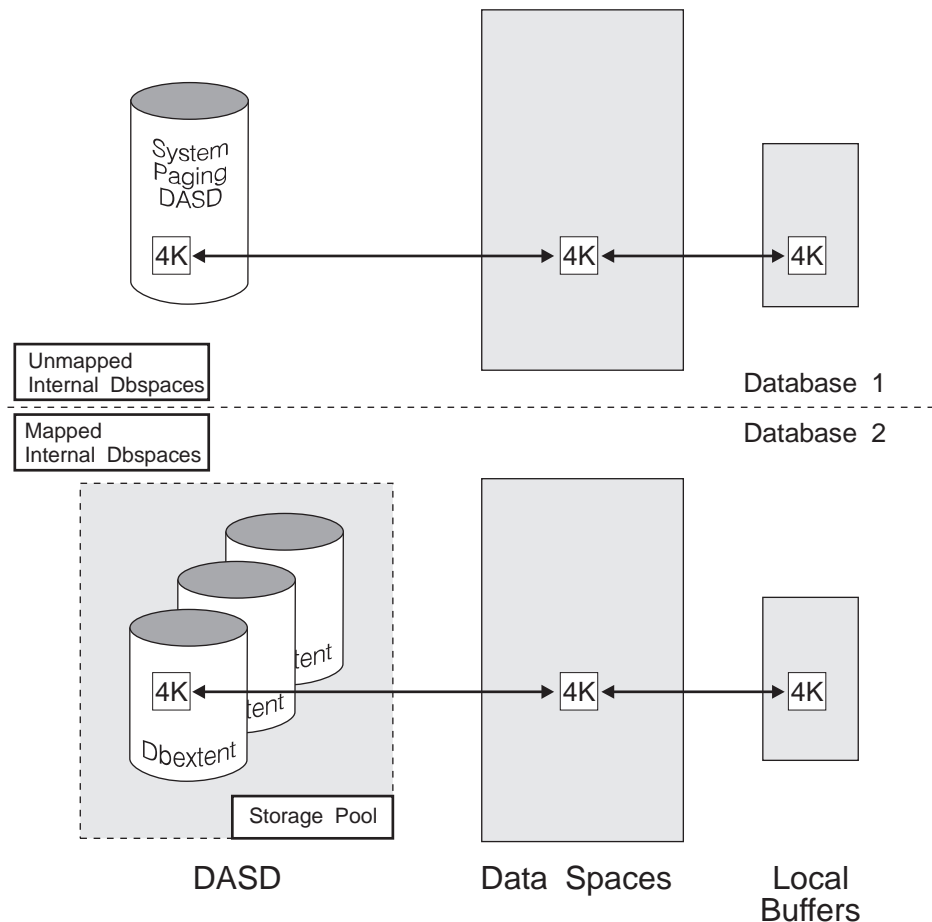


Figure 6. Unmapped and mapped internal dbspaces

The internal dbspaces are still assigned to a storage pool, but they do not use any DASD in that pool. Rather they use system paging DASD. To make up for this, you

must allocate more DASD to system paging. Refer to “VM/ESA Paging DASD” on page 19.

For a description of how to turn on Data Spaces Support for internal dbspaces, refer to “Using Data Spaces with Internal Dbspaces” on page 44. For information on when to use data spaces with internal dbspaces, refer to “Internal Dbspaces” on page 71.

Directory

You can use the *directory* with Data Spaces Support either on or off.

With Data Spaces Support off

The database manager stores directory data on the directory disk (B-disk) in 512-byte blocks and reads these blocks into the *directory* buffers from the B-disk as necessary.

With Data Spaces Support on

The directory uses data spaces as a storage pool would. If the database manager cannot find a block of directory data in the directory buffers, it gets the block from a data space. If the directory block is not already in main storage, the operating system locates the page on the B-disk that contains the directory block and copies it into main storage (each 4KB page contains eight 512-byte directory blocks).

Any DASD accessed through a data space must have a 4KB block size. This means that if you want to use the directory with Data Spaces Support, you must reblock the directory minidisk from 512-byte blocks to 4KB pages.

For information on how to reblock and start Data Spaces Support for the directory, refer to “Using Data Spaces with the Directory” on page 45. For information on when to use Data Spaces Support with the directory, refer to “Directory” on page 72.

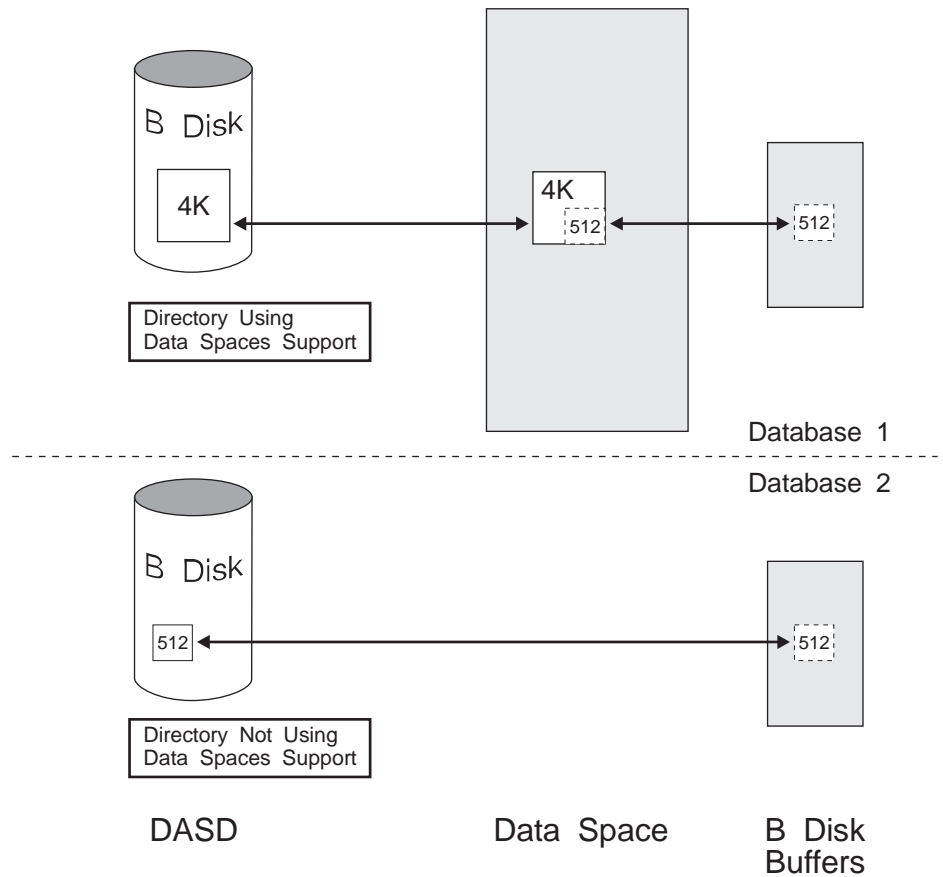


Figure 7. The Database Directory with Data Spaces Support On and Off

Managing Main and Expanded Storage

If unchecked, VMDSS may ask for large amounts of main and expanded storage from the operating system. If it gets it, your database machine may be very fast but other virtual machines in the system may perform poorly.

The operating system always maintains control of its main and expanded storage, and will limit how much your database machine can use. If your machine tries to use more storage than is available, the operating system will release some of your pages (or pages from other virtual machines) from main or expanded storage to make room for the new ones.

Instead of waiting for the operating system to release pages, you can instruct the database manager to do so before your machine reaches its storage limits. Pages to be released will be selected based on the parameters you set. You can also limit the number of modified pages in main and expanded storage to improve checkpoint processing.

The four primary parameters provided are:

- Target working storage size
- Working storage residency priority
- Checkpoint interval
- Save interval.

Target Working Storage Size Parameter

The *target working storage size* parameter controls the amount of main and expanded storage that your database machine uses. When the amount of storage reaches this target, the database manager will start to release certain data space pages immediately after they have been copied into the buffer pools.

While you may exceed this target, the database manager will try to keep you at or below it if possible. (Of course, you may never reach it if the operating system is heavily loaded.)

You can set this parameter at start up time, or dynamically while the database manager is running.

For information on how to change this parameter, refer to “Application Server Initialization Parameters” on page 42. For information on how to choose a value for it, refer to “Choosing the Target Working Storage Size” on page 73.

Working Storage Residency Priorities

The database manager decides which data space pages to release based on the *working storage residency priority* of each pool.

Priorities range from a value of 1 where all pages are released, to a value of 5 where none are. The priorities in between allow a page to be released depending on whether your current working storage size is above or below your target, and whether the page is an index or a data page.

For information on how to change storage priorities, refer to “Specifying Storage Residence Priorities” on page 40. For information on how to choose a storage priority, refer to “Choosing Storage Residence Priorities” on page 73.

The Checkpoint Interval

The checkpoint interval controls the duration between database checkpoints. At a checkpoint, the database manager makes sure that all the modified pages in main and expanded storage are written to DASD. (Refer to “Checkpoints” on page 7.) If there are many modified pages, it can take a long time to complete the checkpoint, and until it is complete the database manager cannot serve any users.

For information on how to change the checkpoint interval, refer to *DB2 Server for VSE & VM Operation*. For information on how to choose a value for it, refer to “Choosing the Checkpoint Interval” on page 75.

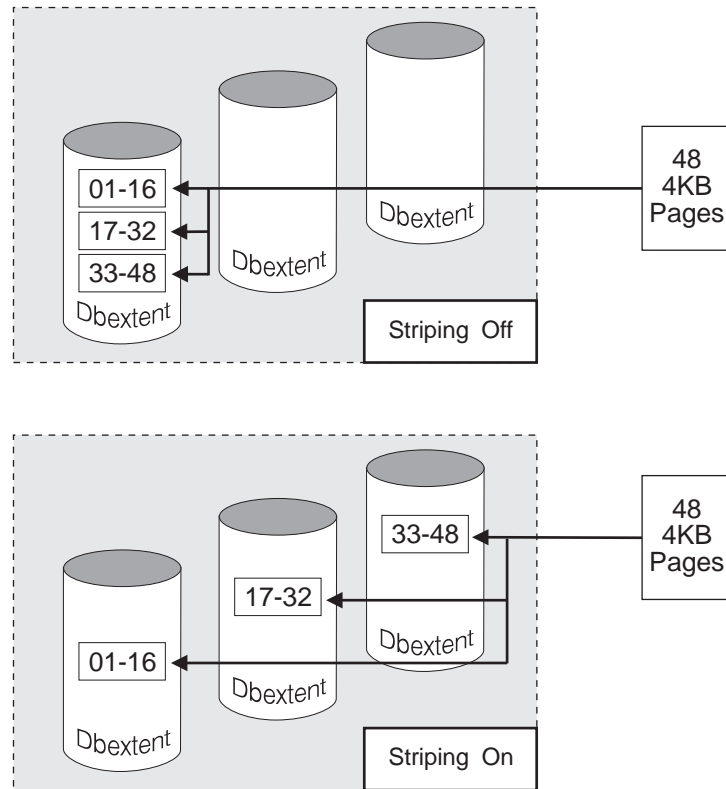
The Save Interval

To make checkpoint processing faster, you can limit the number of modified pages in main and expanded storage by setting the *save interval*. When the number of modified pages in a data space exceeds this parameter, the database manager directs the operating system to save all the modified pages in that data space to DASD. Unlike the save that occurs during checkpoint, the database manager can continue to service users while this is being done.

For information on how to change the save interval, refer to “Application Server Initialization Parameters” on page 42. For information on how to choose a value for it, refer to “Choosing the Save Interval” on page 76.

Striping

VMDS5 will attempt to evenly distribute, or “stripe,” your data across all the dbextents in a storage pool.



Storage Pools

Figure 8. A storage pool with striping switched on and off. The figure shows 48 4KB pages written to DASD with striping on and off.

With striping switched off

The database manager allocates pages in a storage pool in sequence, usually allocating all the pages in one dbextent before using the next dbextent.

With striping switched on

The database manager allocates 16 pages in sequence on each dbextent in the storage pool. The operating system can then read and write the pages to and from DASD in parallel. This may significantly improve DASD performance, depending on how you configure your controllers, channels, and DASD. The optimal configuration would include several dbextents in the storage pool, each on a separate channel, controller, and physical storage device.

For a description of how to use striping, refer to “Turning Striping On and Off” on page 41. For information on how to decide when to use it, refer to “Using Striping” on page 76.

Performance Counters

Several counters are available that can help you monitor the performance of the DASD I/O systems. Each storage pool has its own set of four counters. There is also a set of four counters for unmapped internal dbspaces, and a set for the directory. These counters are different depending on whether a particular component is using data spaces. (Unmapped internal dbspaces always use them.)

For a description of how to display and reset the counters, refer to “Using Storage Pool Performance Counters” on page 48. For more information on the counters, refer to “VMDSS COUNTER POOL Operator Command” on page 61. For information on performance measurements in general, refer to Chapter 4, Measuring Performance.

Planning Structure by Storage Pool

You will need to design the structure of your database so that you can control it effectively.

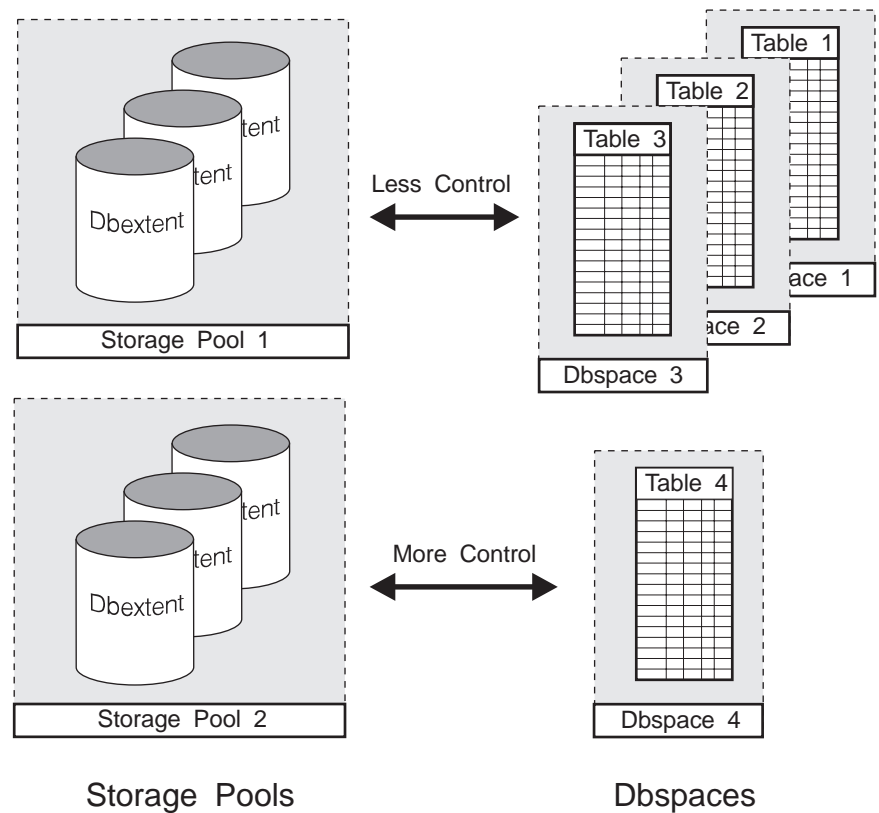


Figure 9. Planning for Critical Tables

We suggest in *DB2 Server for VM System Administration* that you have one table per dbspace. Also, you may want to assign only one dbspace to each storage pool because many of the VMDSS configuration options are grouped by storage pool: for example, you can turn Data Spaces Support and striping on and off for a particular storage pool, set residence priorities by storage pool, and display counters by storage pool. By associating one table per storage pool, you can determine how a specific table will use the VMDSS functions.

However, you should not do this for every table in your database. If you have several tables that you always access together, place each one in a dbspace and assign all the dbspaces to one storage pool. Only special tables where performance and control are critical should have their own storage pool.

Logical and Physical Mapping

When you start the database manager, you can choose whether it will map data space pages to physical pages, *physical mapping*, or virtual pages, *logical mapping* (see below). The type of mapping you choose will apply to all the storage pools in the database that are using Data Spaces Support.

For a description of how to set the type of mapping, refer to “Application Server Initialization Parameters” on page 42. For more information on how to choose one type, refer to “Choosing Logical or Physical Mapping” on page 77.

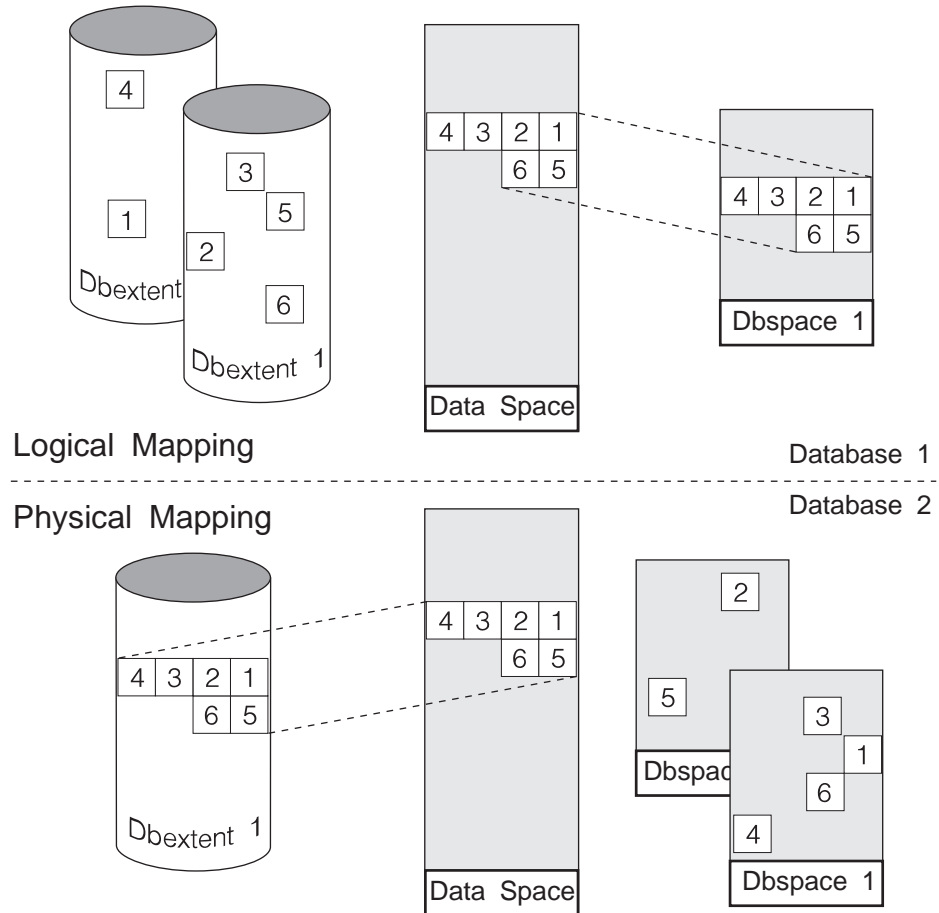


Figure 10. Logical and Physical Mapping

Logical Mapping

This associates (maps) each logical page in a dbspace to a data space page. Since the logical pages are mapped onto a data space in the order they appear in the dbspace, a contiguous set of pages in a data space will correspond to a contiguous series of pages in a dbspace. This is the default and recommended mapping for most applications.

Physical Mapping

This maps each physical page in a dbextent to a data space page. Since the physical pages are mapped onto a data space in the order they appear on DASD, a contiguous series of pages in a data space will correspond to a contiguous series in physical storage.

VSE Guest Sharing

Although VMDSS does not support application servers running in VSE, VSE users can access a VMDSS database in VM/ESA through guest sharing. The database manager runs in one virtual machine, while the VSE users run under a VSE guest system in another virtual machine. Since VMDSS only affects the database manager, VSE guest sharing users will benefit from the same performance improvements as VM users.

For more information on VSE guest sharing refer to the *DB2 Server for VM Program Directory*.

Enabling Requirements

This section describes the operating system, virtual machines, software, virtual storage, and hardware you need to enable and operate VMDSS.

Operating System Overview

To support all of VMDSS's functions, you must:

- Enable it in VM/ESA Version 2 Release 1 (or later)
- Configure your database machine for Extended Configuration (XC) mode.

Operating in Non-XC Mode

If you are not operating in XC mode, you will not be able to use:

- Data Spaces Support for storage pools and the directory
- Unmapped internal dbspaces
- Working storage residency priorities
- Data space performance counters
- The target working storage size parameter

You will be able to use striping, and the storage pool performance counters for the standard DASD I/O system.

Virtual Machine Overview

This section describes the virtual machines you need to enable and operate VMDSS.

MAINT Machine

The MAINT machine, or its equivalent, already exists in all VM systems. It is suggested that you use this machine to update the CP directory, although you can use any machine with write access to the database minidisks and authority to update the CP directory.

SQLMACH Database Machine

The database machine, usually called SQLMACH, owns the database minidisks. It acts as an application server, either locally or remotely, within a TSAF collection or SNA network. For more information, refer to *DB2 Server for VM System Administration*.

To support all VMDSS's functions, you must configure the database machine to operate in XC mode. Refer to "Step 2: Update the CP Directory" on page 25.

You can configure the database machine to operate in ESA mode, but you will then only be able to use a subset of VMDSS's capabilities, as described in "Operating in Non-XC Mode."

Software Requirements

To enable and operate VMDSS, you must first install DB2 Server for VM Version 5 Release 1.

Virtual Storage Requirements

This section describes the virtual storage the MAINT and SQLMACH machines needed to use VMDSS.

MAINT Machine

You do not require any additional virtual storage for the MAINT machine.

SQLMACH Database Machine

You may need to add additional virtual storage to your database machine. To calculate how much:

1. Add 41KB for additional VMDSS code.
2. Add 20KB if you are using data spaces
3. Add 2.5KB for each data space required. Refer to Appendix D, "Determining Number of Data Spaces" on page 91.
4. Add $CUREXTNT \times 16$ bytes. CUREXTNT is the number of dbextents defined during database generation.
5. Add $MAXPOOLS \times 8$ bytes. MAXPOOLS is the maximum number of storage pools that will ever be defined for a database.
6. Add $MAXEXTNT \times 8$ bytes. MAXEXTNT is the maximum number of dbextents that will ever be defined for a database.
7. Add $MAXDBSPC \times 8$ bytes. MAXDBSPC is the maximum number of dbspaces that will ever be defined for a database.

For example, consider a database generated with:

- CUREXTNT = 20
- MAXPOOLS = 256
- MAXEXTNT = 256
- MAXDBSPC = 10240

It is also currently using 10 data spaces for public and private dbspaces. The database machine will use an additional 171KB of virtual storage:

41X1024 =	41,984	
20X1024 =	20,480	
2.5X10X1024 =	25,600	
20X16 =	320	
256X8 =	2,048	
256X8 =	2,048	
<u>+ 10240X8 =</u>	<u>+ 81,920</u>	
174,400 =	174,400 =	170.40KB ~ 171KB

Real Storage Requirements

While you do not require additional real storage (main or expanded storage) to use VMDSS, any storage you add will be used by VMDSS to help improve the performance of your database. Several facilities are included with VMDSS to help you manage how much real storage you use. For more information refer to "Managing Your Working Storage Size" on page 72.

DASD Storage Requirements

This section describes how much DASD space the VM system, the MAINT machine, and the SQLMACH machine need in order to use VMDSS.

Fixed Block Storage Devices

The device number for a minidisk residing on an FBA DASD must start and end on a 4K block boundary. The starting FBA block number and the ending FBA block number +1 of the minidisk must be evenly divisible by 8. Refer to the MAPMDISK information in the *VM/ESA: CP Programming Services* manual.

You can turn Data Spaces Support off for the Storage Pool residing on the FBA device not on a 4K block boundary by updating the storage pool specification file.

To continue using an FBA device, use DDR or use the SQLCDBEX EXEC to copy the extent to a new minidisk that is formatted on a 4K block boundary. Alternately, you can move to a non-FBA device using the SQLCDBEX EXEC. Refer to the *DB2 Server for VM System Administration* for details on how to use the SQLCDBEX EXEC.

VM/ESA Paging DASD

Before you can use separate internal dbspaces (unmapped), you may need to allocate more DASD for VM system paging.

Attention: If VM runs out of system paging DASD, CP will abend if it does not have sufficient pool DASD to accommodate the overflow.

To calculate the maximum number of additional cylinders you need for unmapped internal dbspaces, divide the number of pages currently in all your internal dbspaces by the conversion ratio for your type of DASD, listed in Table 1 and round up to the nearest integer.

Table 1. Additional Paging Cylinders

DASD Type	3350	3375	3380	3390	9345
Conversion Ratio (Blocks per Cylinder)	120	96	150	180	150

For example, if you are currently using 80 internal dbspaces of 1024 pages each for your internal dbspaces, the calculation for 3380 DASD is as follows:

$$80 \times 1024 / 150 = 546.133 \text{ cylinders} \\ \sim 547 \text{ cylinders}$$

Thus, you may need as many as 547 additional 3380 cylinders if you want to use unmapped internal dbspaces. Remember, that the number you calculate will be the maximum you will ever need. While you may actually use far fewer cylinders on a

day to day basis, you still need enough cylinders in either system paging DASD or spool DASD to accommodate your peak requirements. If you cannot supply this maximum value, you can still use mapped data spaces.

To assess your peak requirements, assign your internal dbspaces to their own storage pool and use mapped internal dbspaces, refer to “Mapped Internal Dbspaces” on page 44. You can then use the SHOW POOL operator command (refer to *DB2 Server for VSE & VM Operation*) to see how many pages your production system is really using for internal dbspaces. This will probably be much lower than your maximum calculation. You can then allocate the number of pages the database manager is really using for the internal dspace pool to system paging DASD and start using unmapped internal dbspaces (refer to “Unmapped Internal Dbspaces” on page 44).

Attention: The SHOW POOL command only displays the number of pages a pool is currently using.

You must carefully monitor page usage over a relatively long period until you are confident that the database manager will not use more pages than you will allocate to system paging DASD. Also, remember to continue monitoring SHOW POOL when you start using unmapped internal dbspaces in case your requirements increase.

SQLMACH Database Machine

VMDSS requires a minimum amount of free space on the system minidisks.

System Disks: The system disks are the service and production minidisks or SFS directories of the SQLMACH database machine. No additional DASD is required.

Database Disks: There are three types of database disks:

- Directory
- Log
- Data Extent.

While there is no change to the amount of DASD you require for your log or data extent disks, the DASD you require for the directory disk may change depending on how you use VMDSS.

If you use Data Spaces Support with the directory, you must move the directory from a disk formatted with 512-byte blocks to one with 4KB blocks. Since 4KB blocks use real DASD storage more efficiently than do 512-byte blocks, you do not need as much real DASD storage.

To calculate the number of cylinders you need, multiply the number currently in your directory disk by the conversion ratio for your type of DASD, listed in Table 2 and round up to the nearest integer.

<i>Table 2. Conversion from 512 byte to 4K byte blocks</i>					
DASD Type	3350	3375	3380	3390	9345
Conversion Ratio	0.85	0.63	0.57	0.51	0.52

For example, if you are currently using 34 cylinders of 3380 DASD for your directory disk, the calculation is as follows:

340.57=19.38 cylinders
~20 cylinders

Thus, you will only need a 20 cylinder disk after you move to 4KB pages.

You can also move the directory from a 4KB-block disk to a 512-byte-block disk. To calculate how many cylinders you will then need, divide the number you need when the directory is in 4KB blocks by the conversion and round up to the nearest integer.

Hardware Requirements

To support all of VMDSS's functions you must enable it in a ESA/390 processor within the ES/9000® family that supports VM/ESA in XC mode.

If you enable it into a processor that does not support XC mode, you will only be able to use a subset of its capabilities. Refer to "Operating in Non-XC Mode" on page 17.

Before Enabling

This section describes what you need to read and what decisions you should make before you enable VMDSS.

Program Directory for DB2 Server for VM

Study the *DB2 Server for VM Program Directory* which contains important service information and special instructions.

Preventive Service Planning

Before you enable VMDSS, you should check whether there is any additional Preventive Service Planning (PSP) information that you should know; check with your IBM Support Center or use IBMLINK (ServiceLink).

This program release is maintained through the use of PTF tapes. An updated Version or Release replaces the entire program code; a PTF tape only replaces the changed portion of the program code.

For more information, refer to the *DB2 Server for VM Program Directory*.

Corrective Service

Follow the same corrective service procedures for VMDSS that you follow for DB2 Server for VM. For more information, refer to the *DB2 Server for VM Program Directory*.

Enabling Options

You have several options when you enable VMDSS. Read the following to help you evaluate which one you should use.

Using in a Production System

If you are using VMDSS on an existing production database, you may want to carefully control which, if any, VMDSS functions you use. While the default settings will turn all the functions on (with the exception of Data Spaces Support for the directory), you can reset your operating parameters to turn everything off before you restart your database.

With all the VMDSS functions off, you can ensure that your production system is working as it was before you installed VMDSS. You can then selectively turn on various components (you may need to stop and restart the database manager) and monitor their effect.

Disabling Data Spaces Support

You can move your database manager to an operating system or hardware platform that does not support VMDSS but does support DB2 Server for VM. You may need to do this if you have a backup system that does not meet all of VMDSS's requirements.

Complete the steps listed in “Disabling VMDSS” on page 36; then, move your database manager following your normal procedures.

Resaving DB2 Server for VM in Saved Segments

If you previously stored the DB2 Server for VM DBSS component in a saved segment, you can resave it after you enable VMDSS. Because VMDSS only affects the DBSS component, and does not significantly increase its size (41KB), you can use the default saved segment definition included with the base product. Refer to “Step 10: Resave the DBSS Saved Segment” on page 29 for a description of how to use VMSES/E and the ARISAVES EXEC to resave the DBSS component in a saved segment.

Chapter 2. Enabling

Perform the steps in this chapter to enable the VMDSS code onto the service and production disks and to configure a database machine (SQLMACH) for VMDSS.

Pre-Enable Checklist

Before beginning, make sure that you have completed the following:

- 1. Read Chapter 1, Before You Begin.
- 2. Make sure that you have installed DB2 Server for VM Version 6 Release 1 with at least one 6.1.0 database available. You need this database to verify the enabling of VMDSS.
- 3. Decide whether you will configure the database machine for ESA or XC mode.
- 4. Make sure you have enough space on DASD to complete every step.
- 5. Decide whether you will resave DB2 Server for VM components in saved segments after you enable.
- 6. Read the *DB2 Server for VM Program Directory* to check for any prerequisite Program Temporary Fixes (PTFs) that need to be installed.
- 7. See whether there is any additional Preventive Service Planning (PSP) information that you should be aware of. Check with your IBM Support Center or use IBMLink (ServiceLink).

Operating System Options

This section describes what steps to follow to enable VMDSS to run in either XC mode or non-XC mode, respectively. You need to run in XC mode to support all of VMDSS's functions.

Enabling for XC mode

Perform all the steps summarized on page 23. For "Step 13: Change the Database Administrator Password" on page 32, choose option "B" (Verify XC Mode Installation).

Enabling for non-XC mode

Perform all the steps summarized on page 23 except "Step 2: Update the CP Directory" on page 25. For "Step 13: Change the Database Administrator Password" on page 32, choose option "A" (Verify non-XC Mode Installation).

Enable Checklist

Notes:

- *Perform the steps in order.*
- *Mandatory steps are preceded by squares (■)*
- *Conditional steps are preceded by circles (○)*
- *Page references appear in parentheses.*

- ___ 1. ■ Log onto the MAINT Machine (25)
- ___ 2. ○ Update the CP Directory (25)
- ___ 3. ■ Log off the MAINT Machine (26)
- ___ 4. ■ Log onto the SQLMACH Machine (27)
- ___ 5. ■ Archive your Database (27)
- ___ 6. ■ Activate VMDSS (27)
- ___ 7. ■ Log off the SQLMACH Machine (28)
- ___ 8. ■ Log onto the DB2 for VM Installation User ID (564815A1) (28)
- ___ 9. ■ Link-Edit the Load Library (28)
- ___ 10. ○ Run the Sample Programs to Verify Installation (29)
 - a. ○ Prepare to Build the DB2 for VM Segments (29)
 - b. ○ Build the DB2 for VM Segments (30)
 - c. ○ Create a Bootstrap Package (30)
 - d. ○ Restart the Application Server (32)
- ___ 11. ■ Stop the Application Server (32)
- ___ 12. ■ Add New Users to the Database (32)
- ___ 13. ○ Change the Database Administrator Password (32)
 - a. ○ Verify non-XC Mode Installation (33)
 - b. ○ Verify XC Mode Installation (34)
- ___ 14. ○ System Customization Activities (36)

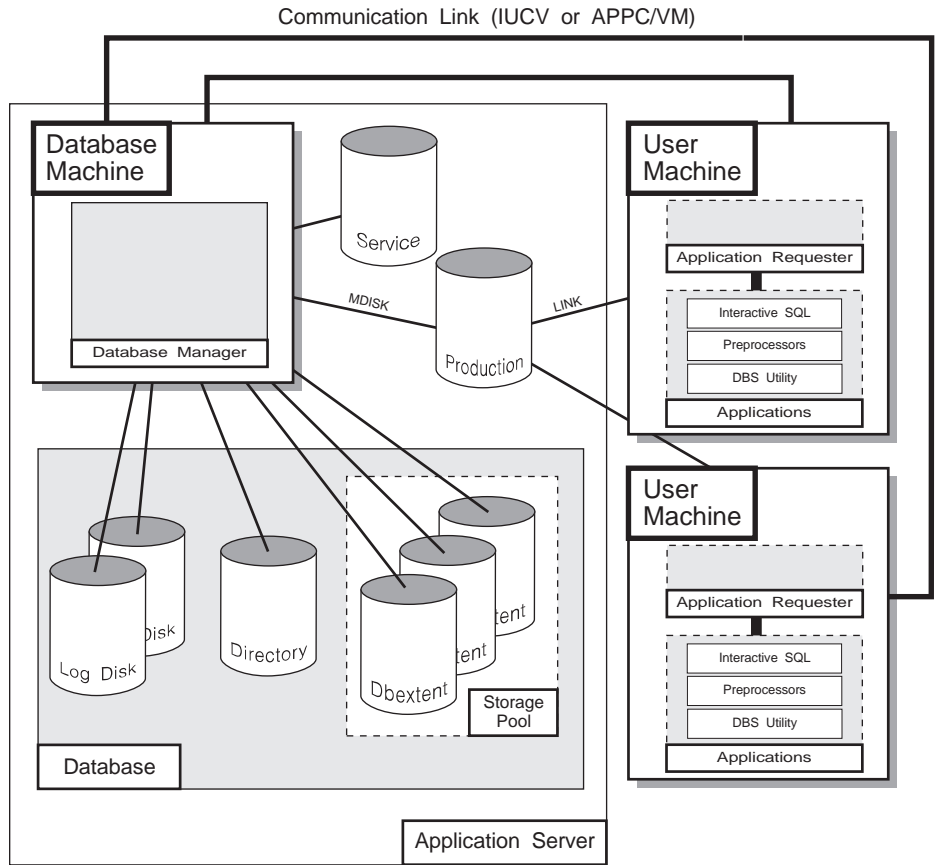


Figure 11. Typical DB2 Server for VM System Setup

Backing Up, Configuring and Enabling Your Database Machine

Perform the following steps to:

- ensure that you have a current backup of your database
- update your database machine for VMDSS
- enable the VMDSS code.

Step 1: Log onto the MAINT Machine

Log onto the MAINT machine.

Step 2: Update the CP Directory

Skip this step if you plan to configure your database machine for ESA mode or if you already updated the CP directory for XC mode when you planned the initial install of DB2 Server for VM.

To use all the VMDSS functions, you must add the statements shown in Figure 12 on page 26 to the CP directory entry for the database machine.


- 
- 1 —> MACHINE XC
 - 2 —> XCONFIG ACCESSLIST ALSIZE 1022
 - 3 —> XCONFIG ADDRSPACE MAXNUMBER 1022 TOTSIZE 2044G

Figure 12. Additional Directory Control Statements for the Database Machine

Statement 1: MACHINE XC

Specifies that the database machine will simulate the VM/ESA XC architecture.

Statement 2: XCONFIG ACCESSLIST ALSIZE 1022

Statement 3: XCONFIG ADDRSPACE MAXNUMBER 1022 TOTSIZE 2044G

The ALSIZE and MAXNUMBER parameters in these two statements specify the maximum number of data spaces that the database machine can create and have existing concurrently. For VMDSS, ALSIZE and MAXNUMBER should be set to the same value.

The value in this example, 1022, is the upper limit for these parameters. Since there is no cost in setting this value high, the value 1022 should be acceptable for most applications. To precisely calculate the maximum number of data spaces that your database machine will use, refer to Appendix D, "Determining Number of Data Spaces" on page 91.

TOTSIZE specifies the maximum total size, in bytes, of all the data spaces that the database machine can create and have existing concurrently.

Since each data space is 2GB, your maximum total size for 1022 data spaces will be 2044GB. For a description of how to precisely calculate the maximum total size of the data spaces for your database, refer to Appendix D, "Determining Number of Data Spaces" on page 91.

For more information on the MACHINE and XCONFIG directory statements, refer to the *VM/ESA: Planning and Administration*.

When you have finished adding the CP directory control statements for the database machine, update the CP directory using your current operating procedures.

Step 3: Log off the MAINT Machine

Log off the MAINT machine.

Step 4: Log onto the SQLMACH Machine

Log onto the database machine (SQLMACH). Refer to “SQLMACH Database Machine” on page 20 for information on this virtual machine.

Step 5: Archive your Database

While installing VMDSS does not affect the data in your database, it is always good practice to archive your database before installing new code or applying service. If you do not archive your database on a regular basis, LOGMODE=Y or N (Y is the default), skip this step.

If your application server is currently running with LOGMODE=L or A, you can perform a user archive or a database archive. To create a database archive, type:

```
SQLEND ARCHIVE
```

As with any archive, the database manager requests that you mount the required tape volume to contain the database archive (or log archive, if LOGMODE=L, and you are not archiving the log to disk). The database manager then creates the archive. When the database manager prompts you to mount and ready the archive volume, you should respond with the virtual device number. Unless you have issued your own CMS FILEDEF command before starting the database manager, the virtual device number for database archives is 181. The virtual device number for log archives is 183.

For more information on the SQLEND command, or information on user archives, refer to *DB2 Server for VM System Administration*.

Step 6: Activate VMDSS

To enable (or remove) the VMDSS code, perform the following steps on the database machine and installation user ID:

1. Be sure you are logged on to the database machine (SQLMACH).
2. Stop the application server using your normal operating procedures.
3. Ensure that the database machine production disk and service disk are linked in write mode. If not, enter:

```
LINK machid 195 195 W  
LINK machid 193 193 W
```

4. Access the production disk with file mode Q and the service disk with file mode V.

```
ACCESS 195 Q  
ACCESS 193 V
```

If you are using SFS directories instead of minidisks, access them with file modes Q and V.

5. Run the ARISDBMA EXEC to identify whether you want DSS code enabled on your production disk. Its syntax is:

```
▶▶ ARISDBMA—DSS (  Y  N ) ▶▶
```

Specify the following parameters:

- Y** Enable the DSS code. This is the default.
- N** Disable the DSS code.

For example, to identify that you want to enable or disable the DSS code, type:

```
ARISDBMA DSS(Y)
ARISDBMA DSS(N)
```

Step 7: Log off the SQLMACH Machine

Log off the database machine.

Step 8: Log onto the DB2 for VM Installation User ID (564815A1)

Log onto the DB2 Server for VM Installation user ID, 564815A1.

Step 9: Link-Edit the Load Library

Rebuild the database manager with VMDSS by link-editing the DBSS component in the ARISQLLD loadlib.

1. Make sure you have read access to the VMSES/E code (MAINT 5E5 disk) and read/write access to the Software Inventory disk (MAINT 51D) or SFS directory.
2. Establish the access order.

```
vmfsetup 5648A70S {DB2VM|DB2VMSFS}
```

5648A70S is the PPF that was shipped with the product. If you have your own PPF override, substitute that name for 5648A70S shown in this command. You also need to substitute your PPF name in the VMSES/E commands in any subsequent steps.

Use DB2VM for installing on minidisks or DB2VMSFS for installing in Shared File System directories.

3. Rebuild DB2 Server for VM DBSS member or component in the ARISQLLD LOADLIB. You must do both steps 3a and 3b.
 - a. Rebuild the ARISQLLD LOADLIB.

```
vmfbld ppf 5648A70S {DB2VM | DB2VMSFS} ARIBLLLD ARISQLDS (a11
vmfview build
```

Use DB2VM for installing on minidisks or DB2VMSFS for installing in Shared File System directories.

ARIBLLLD is the name of the VMSES/E build list used to build the ARISQLLD LOADLIB.

Review the build message log (\$VMFBLD \$MSGLOG). If necessary, correct any problems before you continue. Use the F2 key, ALL, to review all of the messages.

Note: The following message is normal if you are **NOT** running DB2 Server for VM with the DB2 Data Spaces Support:

```
VMFLLB2074I Part xxxxxxx TXT in object ARISQLDS
              in build list ARIBLLLD
              EXEC will be ignored
```

- b. Build the related files.

```
vmfbld ppf 5648A70S {DB2VM | DB2VMSFS} (serviced
vmfview build
```

Use DB2VM for installing on minidisks or DB2VMSFS for installing in Shared File System directories.

Review the build message log (\$VMFBLD \$MSGLOG). If necessary, correct any problems before you continue.

4. Link and access the database machine user ID production and service disks or SFS directories.

```
link SQLMACH 195 295 MR
acc 295 l
link SQLMACH 193 293 MR
acc 293 m
```

You will be prompted for the password to the disks.

Substitute your minidisk addresses, if different.

Substitute in the appropriate SFS directory names.

You also need to substitute your minidisk addresses or SFS directory names in the VMSES/E commands in any subsequent steps.

5. Copy the new ARISQLLD LOADLIB to SQLMACH's production and service disk or directory.

- a. If installing on minidisks, enter the following commands:

```
access 195 i
vmfcopy arisqlld L* i = = l (prodid 5648A70S%DB2VM olddate replace
access 193 j
vmfcopy arisqlld L* j = = m (prodid 5648A70S%DB2VM olddate replace
```

The VMFCOPY command updates the VMSES PARTCAT file on the production disk (195) and the service disk (193).

- b. If installing using Shared File System, enter the following commands:

```
access 5648A70S.sql.production i
access SQLMACH.sql.production l
vmfcopy arisqlld L* i = = l (prodid 5648A70S%DB2VM olddate replace
access 5648A70S.sql.service j
access SQLMACH.sql.service m
vmfcopy arisqlld L* j = = m (prodid 5648A70S%DB2VM olddate replace
```

The VMFCOPY command updates the VMSES PARTCAT file.

Step 10: Resave the DBSS Saved Segment

If you are using a saved segment for DBSS then you need to resave it. Use the following steps to resave the saved segment, otherwise continue with Step 11.

Step 10a. Prepare to Build the DB2 Server for VM Segments

Before building the new DB2 Server for VM segment, following these steps:

1. Clear your virtual machine by entering the following IPL command. This command bypasses the execution of the system profile (SYSPROF EXEC) and without loading the installation saved segment (CMSINST).

```
ipl cms parm clear nosprof instseg no
```

Note: **** DO NOT press ENTER at the VM READ!****

2. Bypass the execution of the PROFILE EXEC by entering the following command:
`access (noprof`
3. Access the VMSES/E code by entering the following command:
`access 5e5 b`
4. Link and access the Software Inventory disk by entering the following commands:
`link MAINT 51d 51d mr`
`access 51d d`
5. Access the database machine, SQLMACH, production minidisk or SFS directory by entering the following command:
`access vdev k`

vdev is the address the database machine production minidisk is linked as by the installation user ID, or *vdev* is the name of the database machine production SFS directory. You need write access to this minidisk or directory.
6. Before running the VMFBLD command to save the segments, activate the user language files by entering the following CMS command:
`set language ameng (add ari user`
7. Release the database machine, SQLMACH, production minidisk or SFS directory by entering the following command:
`rel k`

Step 10b. Build the DB2 Server for VM Segments

To build the DB2 Server for VM segments, enter the following command:

```
vmfbld ppf segbld esasegs segblist SQLSQLDS (serviced
```

If you are using a different name for the DBSS saved segment substitute your name in place of SQLSQLDS in the VMFBLD command. The ARISAVES is called by the VMFBLD command.

Step 10c. Create a Bootstrap Package

If you responded YES when prompted by the ARISAVES EXEC to use the saved segments you that loaded as defaults, you do not have to do this step, as ARISAVES would have generated a default bootstrap package (SQLDBA) for you. Continue with “Step 11: Stop the Application Server” on page 32.

If you answered NO to the prompt, you must run the SQLGENLD EXEC to create a bootstrap package for the saved segments you loaded. To run SQLGENLD EXEC, you must log off of the installation user ID and log on to the database machine.

Because SQLGENLD prompts you for certain information about the new bootstrap, you should determine the contents of the bootstrap package before you run the SQLGENLD EXEC. For more information, see “Contents of a Bootstrap Package.”

Contents of a Bootstrap Package: A bootstrap package contains modules created by the SQLGENLD EXEC. SQLGENLD places the modules on the production minidisk (Q-disk). Note that, even though the DBSS and RDS components are loaded in different saved segments, there is only one bootstrap module for them. All of those components are needed to run the DB2 Server for

VM system code in a database machine. Thus, one bootstrap identifies the location of the DBSS and RDS components.

Not all modules are needed because the database manager uses defaults when a module of a bootstrap is missing. For more information on the defaults, see "Using SQLGENLD."

Figure 13 summarizes the different bootstrap modules that you can have.

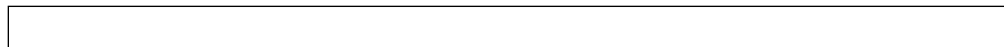


Figure 13. Bootstrap Package Contents

The *dcssid* (saved segment ID) is the name you give to the bootstrap package with SQLGENLD. It is the *dcssid* that you use in the DCSSID parameter of various IBM-supplied execs (such as, SQLSTART or SQLINIT). When *dcssid* is specified in a DCSSID parameter, the bootstrap package production disk entries are copied to the execution machine's A-disk as shown in Figure 14.

Production Q-disk Entry				Execution Machine A-disk Entry		
FN	FT	FM	COPY/RENAME	FN	FT	FM
<i>dcssid</i>	SQLRMBT	Q	TO	ARISRMBT	MODULE	A
<i>dcssid</i>	SQLDBBT	Q	TO	ARISDBBT	MODULE	A
<i>dcssid</i>	SQLISBT	Q	TO	ARISISBT	MODULE	A

Figure 14. Bootstraps Copied to the Execution Machine A-disk

The resource adapter bootstrap is incomplete when it is copied to the A-disk of the user machine. It is completed when the user runs the SQLINIT EXEC, which supplies the missing server name to be accessed.

Use SQLGENLD to generate bootstrap packages for running the database manager in saved segments. You cannot use this EXEC to generate a bootstrap package for running the database manager in a default mode. The SQLDBA bootstrap package identifies the default mode, which can be default saved segments (if you have defined them) or user free storage.

Using SQLGENLD: When you identify the bootstraps to be contained in the package you are creating and the location where you want them to load the code, you can use the SQLGENLD EXEC. To use SQLGENLD, obtain read access to the service minidisk by entering the following command:

```
access 193 v
```

You can run SQLGENLD only from the database machine:

```
sqlgenld
```

When it runs, the SQLGENLD EXEC obtains both read and write access to the production minidisk. Both kinds of access are available to a defined database machine. You should ensure that no other machine has write access to the production minidisk when you run SQLGENLD.

If you are running SQLGENLD from a database machine that does not own the production minidisk, SQLGENLD prompts you for the write password.

The SQLGENLD EXEC prompts you for *dcssid*. This is the name of the new bootstrap package. If a bootstrap package with this name already exists, SQLGENLD replaces the existing bootstraps. The EXEC does not let you replace the initial SQLDBA bootstrap package. The SQLDBA bootstrap package is used as a default by many IBM-supplied execs. Do not modify or erase the SQLDBA bootstrap package.

When you supply *dcssid*, SQLGENLD prompts if you want to create a resource adapter bootstrap, a DBSS/RDS bootstrap, and an ISQL bootstrap. For each bootstrap that you choose to create, you are prompted for the saved segment name (or, in the case of DBSS/RDS, names). The name is the name you used in the DEFSEG command.

The database manager prompts if you want this bootstrap package to be the default DCSSID for user machines that have a link to this production (Q) disk. Specify this as the default if you have users linking to this Q-disk who will be accessing a database machine that does not own this production (Q) disk, and if you do not have saved segments identified by the SQLDBA bootstrap package. Because the database manager provides a default DCSSID, these users are not required to specify the DCSSID parameter when they run the SQLINIT EXEC.

Note: The SQLDCSID DEFAULT file cannot be used by a user if the file *resid* SQLDBN exists on the production (Q) disk they are linked to. This is because the default bootstrap package for a database is identified in the *resid* SQLDBN file. The SQLDCSID DEFAULT file is used by users that are accessing an application server other than the one that owns the Q-disk to which they are linked.

If you say that you want this bootstrap to be the default for users with a link to this production (Q) disk, a new file SQLDCSID DEFAULT will be created on the production (Q) disk to contain the default DCSSID. When the bootstraps are created, SQLGENLD places them on the production minidisk. They are then erased from the database machine A-disk.

Step 10d. Restart the Application Server

Restart the application server in multiple user mode with the required PROTOCOL parameter.

Step 11: Stop the Application Server

Log off the DB2 Server for VM installation machine (if not already done).

Step 12: Add New Users to the Database

Log onto the SQLMACH machine (if not already done).

Step 13: Change the Database Administrator Password

You must now verify that you enabled the VMDSS code successfully.

Perform either Step 13A or 13B, depending on whether you chose to enable in non-XC mode or XC mode.

Step 13A: Verify non-XC Mode Installation

Check that your database machine is not in XC mode by typing `#cp query set`. For example:

```
#cp query set
cp query set
MSG ON , WNG ON , EMSG ON , ACNT OFF, RUN OFF
LINEDIT ON , TIMER ON , ISAM OFF, ECMODE ON
ASSIST OFF , PAGEX OFF, AUTOPOLL OFF
IMSG ON , SMSG ON , AFFINITY NONE , NOTRAN OFF
VMSAVE OFF, 370E OFF
STBYPASS OFF , STMULTI OFF 00/000
MIH OFF , VMCONIO OFF , CPCONIO OFF , SVCACCL OFF , CONCEAL OFF
MACHINE XA, SVC76 CP, NOPDATA OFF, IOASSIST OFF
CCWTRAN ON
```

If you are in XC mode (MACHINE XC), go to “Step 2: Update the CP Directory” on page 25 and check that the CP directory entries listed in that step are not included in your database machines directory entries.

Start the application server in multiple user mode by entering:

```
SQLSTART DB(server_name)
```

Replace *server_name* with the name of your database. This name is specified in the IUCV *IDENT statement contained in the CP directory for the database machine. For more information on the SQLSTART command, refer to the *DB2 Server for VSE & VM Operation* manual.

For example:

```
sqlstart db(sqldba)
ARI0717I Start SQLSTART EXEC: 05/28/91 11:05:52 EDT.
ARI0320I The default database name is SQLDBA.
:
ARI2015I The storage pool specification input file was not
found. The database manager will use the default values.
ARI2020I The machine is not in XC-mode.
Data spaces will not be used.
ARI2027I No storage pools will use data spaces.
ARI0283I Log analysis is complete.
ARI0282I LUW UNDO is completed.
ARI0281I LUW REDO is completed.
ARI0143I The application server has been initialized
with the following values:
CHARNAME = INTERNATIONAL, DBCS = NO, CHARSUB = SBCS,
CCSIDSBBCS = 500, CCSIDMIXED = 0, CCSIDGRAPHIC = 0.
ARI0134I Application server FTMACH6 has been
identified as a global resource.
ARI0060I database manager initialization complete.
ARI0045I Ready for operator communications.
```

Note, the underlined messages show that VMDSS is installed. Because the database machine is not in XC mode, the database manager will use the standard DASD I/O system instead of Data Spaces Support. To confirm this, type **counter pool 1**. You should see something like the following:

```

counter pool 1
Counter values at DATE='05-28-91' TIME='11:14:35'.

Pool No. 1: *BLOCKIO
Pages looked at in the buffer LBUFL00K: 121
Page reads PGREAD : 26
Page writes PGWRITE : 0
IUCV *BLOCKIO I/O requests IUCVBIO : 26
ARI0065I Operator command processing is complete.

```

If *BLOCKIO appears, this tells you that the database manager is using the standard DASD I/O system for storage pool 1.

While you cannot use data spaces in non-XC mode, you can use striping. Since it is the default setting for the database manager to use striping, you should see something like the following if you type **show pool 1**:

```

show pool 1

POOL NO. 1: NUMBER OF EXTENTS = 2 BLK STR

EXTENT TOTAL NO. OF NO. OF NO. OF %
NO. PAGES PAGES USED FREE PAGES RESV PAGES USED
1 855 77 778 9
2 855 0 855 0
TOTAL 1710 77 1633 20 7
ARI0065I Operator command processing is complete.

```

If STR appears, this tells you that the database manager is using striping for storage pool 1.

Continue with “Step 14: System Customization Activities” on page 36.

Step 13B: Verify XC Mode Installation

Check that your database machine is in XC mode by typing **#cp query set**. For example:

```

#cp query set
CP QUERY SET
MSG ON , WNG ON , EMSG ON , ACNT OFF, RUN OFF
LINEDIT ON , TIMER OFF , ISAM OFF, ECMODE ON
ASSIST OFF , PAGEX OFF, AUTOPOLL OFF
IMSG ON , SMSG ON , AFFINITY NONE , NOTRAN OFF
VMSAVE OFF, 370E OFF
STBYPASS OFF , STMULTI OFF 00/000
MIH OFF , VMCONIO OFF , CPCONIO OFF , SVCACCL OFF , CONCEAL OFF
MACHINE XC , SVC76 CP, NOPDATA OFF, IOASSIST OFF
CCWTRAN ON

```

If you are not in XC mode, return to “Step 2: Update the CP Directory” on page 25 and check the CP directory entries for your database machine.

Start the application server in multiple user mode by entering:

```
SQLSTART DB(server_name)
```


Note: A storage pool used only for internal dbspaces and which has a dbextent on a virtual disk cannot be used with data spaces turned on for that pool. This storage pool **must** be specified with the BLK and SEQ options in the storage pool specification file. See Appendix B, "Storage Pool Specification File Format" on page 85.

Replace *server_name* with the name of your database. This name is specified in the IUCV *IDENT statement contained in the CP directory for the database machine. For more information on the SQLSTART command, refer to the *DB2 Server for VSE & VM Operation* manual.

For example:

```
sqlstart db(sqldba)
ARI0717I Start SQLSTART EXEC: 05/28/91 10:51:06 EDT.
ARI0320I The default database name is SQLDBA.
:
ARI0015I SEPINTDB parameter value is Y.
ARI0016I SAVEINTV parameter value is 10.
ARI0015I MAPPING parameter value is L.
ARI0016I TARGETWS parameter value is 32.
ARI2015I The storage pool specification input file was not
found. The database manager will use the default values.
ARI2026I Some or all storage pools will use data spaces.
ARI0283I Log analysis is complete.
ARI0282I LUW UNDO is completed.
ARI0281I LUW REDO is completed.
ARI0143I The application server has been initialized
with the following values:
CHARNAME = INTERNATIONAL, DBCS = NO, CHARSUB = SBCS,
CCSIDBCS = 500, CCSIDMIXED = 0, CCSIDGRAPHIC = 0.
ARI0134I Application server FTMACH6 has been
identified as a global resource.
ARI0060I database manager initialization complete.
ARI0045I Ready for operator communications.
```

Note, the underlined messages show that VMDSS is installed and, because the database machine is in XC-mode, the database manager is using Data Spaces Support. To confirm this, type **counter pool 1**. You should see something like the following:

```
counter pool 1
Counter values at DATE='05-28-91' TIME='10:59:24'.

Pool No. 1: Data Spaces
Pages looked at in the buffer LBUFL00K: 121
Pages moved from DS to buffer DSREAD : 26
Pages moved from buffer to DS DSWRITE : 0
DS page fault notifications DSFAULT : 6
ARI0065I Operator command processing is complete.
```

If Data Spaces appears, this tells you that the database manager is using Data Spaces Support for storage pool 1.

Step 14: System Customization Activities

You may now choose any of the following options:

- Create a new database to use Data Spaces Support (refer to “Using Data Spaces Support with a New Database” on page 48).
- Reblock the directory disk of an existing database to use Data Spaces Support (refer to “Using Data Spaces with the Directory” on page 45).
- Change the VMDSS storage pool specifications (refer to “Storage Pool Specifications” on page 39). These specifications turn Data Spaces Support on and off, set storage residency priorities, and turn striping on and off.
- Change the VMDSS initialization parameters (refer to “Application Server Initialization Parameters” on page 42). These parameters set your application server’s save interval, target working storage, and whether it will use mapped or unmapped internal dbspaces.

Disabling VMDSS

If you want to disable VMDSS from your service and production disks perform the Steps 1-4 and 6 in the database machine (SQLMACH), and Step 5 from the installation user machine.

Disable Step 1: Archive your Database

If you regularly archive your database, type the following at the operator console:

```
SQLEND ARCHIVE
```

For more information on the SQLEND command, refer to “Step 2: Update the CP Directory” on page 25 or see the *DB2 Server for VSE & VM Operation* manual.

Disable Step 2: Access the Service Disk or Directory

Accesses the DB2 Server for VM service minidisk with file mode V.

```
ACCESS 193 V
```

If you are using a service SFS directory instead of a minidisk, access it with file mode V.

Disable Step 3: Reblock the Directory Disk

Your directory disk must be formatted with a block size of 512-bytes in order for the database manager to be able to use it without VMDSS. If it is formatted with the 4KB size, you must reblock it. Follow the instructions in “Reblocking the Database Directory” on page 45.

Disable Step 4: Remove the VMDSS Files

To remove the VMDSS files you need to run ARISDBMA with the DSS(N) option. See “Step 6: Activate VMDSS” on page 27.

Disable Step 5: Link-Edit the Load Library

Rebuild the database manager without VMDSS by link-editing the DBSS component. To do the link-edit on the load library follow “Step 8: Log onto the DB2 for VM Installation User ID (564815A1)” on page 28 through “Step 10: Resave the DBSS Saved Segment” on page 29. These steps will include the rebuilding of the DBSS saved segment.

Disable Step 6: Restart the Application Server

Start the application server in multiple user mode using your normal operating procedures.

Chapter 3. Operating

This section describes how to complete the tasks associated with operating and customizing VMDSS.

Storage Pool Specifications

There are three VMDSS specifications that you can set for storage pools:

- Whether Data Spaces Support or the standard DASD I/O system is used
- The working storage residency priority, for those pools that use Data Spaces
- Whether or not striping is used.

The default settings are that every storage pool will use data spaces, a working storage residency priority of 3, and striping.

Note: A storage pool used only for internal dbspaces and which as a dbextent on a virtual disk cannot be used with data spaces turned on for that pool. This storage pool **must** be specified with the BLK and SEQ options in the storage pool specification file. See Appendix B, "Storage Pool Specification File Format" on page 85.

You can change these settings either at database startup, or (except for the first one) dynamically while the database is running.

Changing Storage Pool Specifications at Startup

To change the storage pool specifications of your database at startup, you need to create a *storage pool specification file*. You can read the next few sections to learn how to do this, or you can refer to Appendix B, "Storage Pool Specification File Format" on page 85 for a summary of the file's syntax.

At startup, the application server looks for the storage pool specification file. It should have a file name that corresponds to your database's *server_name*, a file type of ARISPOOL and a file mode of *.

If you want to use a different file name or file type, enter a CMS FILEDEF command to identify a file as the storage pool specification file. For example:

```
FILEDEF ARISPOOL DISK SPSPEC FILE A
```

where SPSPEC FILE A identifies the storage pool specification file. The FILEDEF syntax is:

►—FILEDEF ARISPOOL DISK—*filename*—*filetype*—*filemode*—◄

filename filetype filemode

Specifies the file name, file type, and file mode of the storage pool specification file.

Add your specifications to the specification file as described below, and start the application server. If you want to add or change any specifications, you must:

1. Stop the application server (SQLEND)
2. Update the storage pool specification file
3. Restart the application server (SQLSTART)

Specifying Either Data Spaces Support or Standard DASD I/O

To change this setting for a particular storage pool, add a line to the specification file to specify either Data Spaces Support (DS) or standard DASD I/O (BLK). (DS is the default.)

For example, consider a database with five storage pools. To use Data Spaces Support for storage pool 1 and standard DASD I/O for pools 2 to 5, your specification file would look like:

```
-- Storage Pool Specification File

1 DS -- This line turns on Data Spaces Support for pool 1
2-5 BLK -- This line turns off Data Spaces Support
      -- for pools 2 to 5
```

The text is optional comments. If you add any comments, precede them by two dashes.

Note that DS is the default parameter, so you can also code the file like:

```
-- Storage Pool Specification File

2-5 BLK -- This line turns off Data Spaces Support
      -- for pools 2 to 5
```

For information on when to use data spaces with storage pools, refer to “Storage Pool” on page 71.

Specifying Storage Residence Priorities

To set the storage residence priority of a storage pool that uses Data Spaces Support, add an integer (from 1 to 5) to the end of the DS parameter in your specification file. (3 is the default.)

For example, to use priority 1 with pools 3 and 4, priority 3 with pool 2, and priority 4 with pool 5, your specification file would look like:

```
-- Storage Pool Specification File

1 BLK -- This line turns off Data Spaces Support for pool 1
2 DS -- This line uses residency priority 3 for pool 2
3-4 DS1 -- This line uses residency priority 1 for pools 3 and 4
5 DS4 -- This line uses residency priority 4 for pool 5
```

Pool 1 is not using Data Spaces Support, so it is not assigned any priority. Pool 2 is using the default value, so the integer 3 does not have to be included.

For a description of the five priorities and how to choose one, refer to “Choosing Storage Residence Priorities” on page 73.

Turning Striping On and Off

To turn striping on for a particular storage pool, add the three-letter code STR to the end of the line for that pool. To turn it off, add the code SEQ.

For example, to turn striping on for storage pools 1, 3, and 5, and to turn it off for pools 2 and 4, your specification file would look like:

```
-- Storage Pool Specification File

1  BLK STR  -- Striping turned on
2  DS  SEQ  -- Striping turned off
3  DS1 STR  -- Striping turned on
4  DS1 SEQ  -- Striping turned off
5  DS4      -- Striping left on by default
```

For information on when to use striping, refer to “Using Striping” on page 76.

Checking Your Current Storage Pool Specifications

You can display your current storage pool specifications from the operator console, or through ISQL, with the SHOW POOL operator command.

For example, the following screen shows you that pool 2 is using Data Spaces Support with a storage residency priority of 3, and striping:

```
show pool 2

POOL NO.  2:      NUMBER OF EXTENTS = 6  DS3 STR

EXTENT  TOTAL  NO. OF  NO. OF  NO. OF  %
NO.     PAGES  PAGES USED  FREE PAGES  RESV PAGES  USED
  1     200070    55821   144249             27
  2     200070    54645   145425             27
  3     200070    56965   143105             28
  4     200070    56336   143734             28
  5     200070    55210   144860             27
  6     200070    56267   143803             28
TOTAL  1200420   335244   865176             20  27
ARI0065I Operator command processing is complete.
```

For a detailed description of this command, refer to the *DB2 Server for VSE & VM Operation* manual.

Changing Storage Pool Specifications Dynamically

Once the application server is running, if you want to change the setting for Data Spaces Support (DS or BLK) you must shut down and restart the database manager as described in “Changing Storage Pool Specifications at Startup” on page 39. However, the two other specifications—working storage residency priority and striping— can be changed dynamically by issuing the SET POOL command from the operator console. For example:

```

set pool 1 ds2 seq
ARI0065I Operator command processing is complete.
show pool 1

POOL NO. 1:      NUMBER OF EXTENTS = 2  DS2_SEQ

EXTENT  TOTAL  NO. OF  NO. OF  NO. OF  %
NO.     PAGES  PAGES  USED   FREE   PAGES  RESV  PAGES  USED
  1      855    74     781           8
  2      855    47     808           5
TOTAL   1710   121    1589    20     7
ARI0065I Operator command processing is complete.

```

Note: Any changes you make using the SET POOL command are only in effect while the application server is running. If you stop and restart the application server, it will use the settings in the storage pool specification file, which are unchanged by the SET POOL command.

For a detailed description of the SET POOL command, refer to the *DB2 Server for VSE & VM Operation* manual.

Application Server Initialization Parameters

There are four VMDSS parameters that can be added to the application server startup EXEC (SQLSTART):

MAPPING = L|P

Set whether the database manager will use logical (L) or physical (P) mapping.

SAVEINTV = n

Sets the save interval for the database to *n* blocks of 4KB pages (32 pages in a block).

SEPINTDB = Y|N

Sets whether the database manager will use separate unmapped internal dbspaces (Y) or not (N).

TARGETWS = n

Sets the target working storage size for the database to *n* megabytes.

For a brief description of the VMDSS parameters and their default values, refer to Appendix C, "DB2 Server for VM Initialization Parameters" on page 89. For a detailed description of the standard SQLSTART EXEC, refer to *DB2 Server for VSE & VM Operation*. For information on how to choose values for these parameters, refer to Chapter 5, "Tuning Performance" on page 69.

Changing Initialization Parameters at Startup

To set any of these parameters at startup, run the SQLSTART EXEC following your current operating procedures, and include the parameters you want. For example:


```

sqlstart db(SQLDBA) parm(mapping=L saveintv=12 sepintdb=y targetws=30)
ARI0717I Start SQLSTART EXEC: 05/23/91 09:48:57 EDT.
ARI0320I The default database name is SQLDBA.
:
ARI0015I SEPINTDB parameter value is Y.
ARI0016I SAVEINTV parameter value is 12.
ARI0015I MAPPING parameter value is L.
ARI0016I TARGETWS parameter value is 30.
:

```

Displaying TARGETWS and SAVEINTV

You can display the current save interval and the current target working set size with the SHOW SAVEINTV and the SHOW TARGETWS operator commands respectively. (There is no SHOW SEPINTDB or SHOW MAPPING command.) For example:

```

show targetws
TARGETWS = 32 MB
Current Working Storage = 36 MB
                        (31 MAIN + 5 EXPANDED)
ARI0065I Operator command processing is complete.
show saveintv
SAVEINTV = 10
ARI0065I Operator command processing is complete.

```

For a detailed description of these commands refer to the *DB2 Server for VSE & VM Operation* manual.

Changing TARGETWS and SAVEINTV Dynamically

You can dynamically change the save interval and the target working set size from the operator console with the SET SAVEINTV and the SET TARGETWS commands respectively. For example:

```

set targetws 30
Previous TARGETWS = 32 MB
TARGETWS = 30 MB
Current Working Storage = 36 MB
                        (31 MAIN + 5 EXPANDED)
ARI0065I Operator command processing is complete.
set saveintv 12
Previous SAVEINTV = 10
SAVEINTV = 12
ARI0065I Operator command processing is complete.

```

Note: Any changes you make using SET TARGETWS or SET SAVEINTV are only in effect while the application server is running. If you stop and restart the application server, it will use the settings you specify in the SQLSTART command.

For a detailed description of these commands refer to the *DB2 Server for VSE & VM Operation* manual.

Using Data Spaces with Internal Dbspaces

This section describes how to use internal dbspaces with data spaces. For information on whether these dbspaces should be mapped or unmapped, refer to “Internal Dbspaces” on page 71.

Unmapped Internal Dbspaces

To use separate (unmapped) internal dbspaces, set the initialization parameter SEPINTDB to “Y” (Y is the default). For example:

```
sqlstart db(SQLDBA) parm(sepintdb=Y)
ARI0717I Start SQLSTART EXEC: 05/23/91 09:44:24 EDT.
ARI0320I The default database name is SQLDBA.
:
ARI0015I SEPINTDB parameter value is Y.
:
```

Message ARI0015I should tell you that the SEPINTDB parameter is set to Y. If it does not, check that you are operating your database machine in XC mode. You cannot use unmapped internal dbspaces in ESA mode.

Before you use unmapped internal dbspaces, you must allocate more DASD to VM system paging. Refer to “VM/ESA Paging DASD” on page 19.

Attention: If VM runs out of system paging DASD, CP will abend if it does not have sufficient spool DASD to accommodate the overflow.

Mapped Internal Dbspaces

To use mapped internal dbspaces, turn Data Spaces Support on for the storage pool containing internal dbspaces and set the initialization parameter SEPINTDB to “N.” For example:

```
sqlstart db(SQLDBA) parm(sepintdb=N)
ARI0717I Start SQLSTART EXEC: 05/23/91 09:39:02 EDT.
ARI0320I The default database name is SQLDBA.
:
ARI0015I SEPINTDB parameter value is N.
:
```

Note, just setting SEPINTDB=N does not turn Data Spaces Support on or off for your internal dbspaces. You can use either Data Spaces Support or the standard DASD I/O system with internal dbspaces. Because internal dbspaces are assigned to one storage pool, they will use whichever DASD I/O system is specified for that pool (see “Specifying Either Data Spaces Support or Standard DASD I/O” on page 40).

Note: A storage pool used only for internal dbspaces and which as a dbextent on a virtual disk cannot be used with data spaces turned on for that pool. This storage pool **must** be specified with the BLK and SEQ options in the storage pool specification file. See Appendix B, “Storage Pool Specification File Format” on page 85.

Using Data Spaces with the Directory

You can use the directory with either Data Spaces Support or the standard DASD I/O system.

To use Data Spaces Support, format your directory disk with a block size of 4096 bytes (4KB). The database manager will automatically use data spaces when it detects the 4KB blocks. If the directory disk is formatted with a 512-byte block size, the standard I/O system will be used instead.

If your directory disk is currently formatted for 512-byte blocks and you want to use Data Spaces Support, you can reblock your disk with the SQLCDBEX EXEC (refer to “Reblocking the Database Directory”).

If the database manager is using Data Spaces Support with the directory, you will see the following message at startup time:

```
⋮
ARI2022I the database manager is using data spaces for the directory.
⋮
```

You can also check this information by displaying the storage pool counters for the directory. For example:

```
counter pool dir
Counter values at  DATE='05-23-91'  TIME='15:41:07'

Directory:  Data Spaces
Pages looked at in the buffer  LBUFL00K:  21
Pages moved from DS to buffer  DSREAD  :  44
Pages moved from buffer to DS  DSWRITE :  27
DS page fault notifications    DSFAULT :  4
ARI0065I Operator command processing is complete.
```

For information on when to use data spaces with the directory, refer to “Directory” on page 72.

Reblocking the Database Directory

The SQLCDBEX EXEC is updated for VMDSS, and now asks you which block size you want the output directory to be. For a block size of 512 bytes, type **512**; for a block size of 4KB bytes, type **4096**. If you do not type in a size and just press Enter, the EXEC will make the output directory disk the same block size as the input directory disk.

For more information on the standard SQLCDBEX EXEC refer to *DB2 Server for VM System Administration*. For a detailed description of the updated VMDSS version refer to Appendix A, “EXECs” on page 79.

Note that if you reblock the directory from 512-bytes blocks to 4KB blocks, you will not need as much DASD storage on the new directory disk; if you reblock from 4K to 512, you will need more. To calculate the number of cylinders you will need, refer to “Database Disks” on page 20.

If you plan to switch between blocking sizes often, you may want to keep one disk reserved for the 512-byte blocked directory and one for the 4KB directory. You can define each disk to the appropriate size for its blocking, and copy the directory back and forth between disks.

If you plan to move from a 512-byte to a 4KB disk of the same size, you can take advantage of the 4KB blocking by expanding the directory to fit the new disk. However, if you need to return to a 512-byte disk, you will have to copy it back to a larger disk.

(When the SQLCDBEX EXEC finds that there is more room on a new disk than it needs for the current directory, it will ask you whether it should expand the directory to fit the new disk. If you tell it not to expand the directory, you cannot take advantage of the unused portion of the new disk.)

Example

Consider a database with a directory disk (B-disk) at address 300 and a block size of 512 bytes. To reblock the B-disk to 4096 bytes, run the SQLCDBEX EXEC to copy the directory onto a new disk (305) blocked to 4KB as shown in the following example:

sqlcldbex db(SQLDBA)

ARI0717I Start SQLCDBEX EXEC: 05/23/91 08:58:36 EDT.

ARI6102A Enter DBEXTENT number (or LOGDSK1, LOGDSK2,
or BDISK) to copy.
(Enter a null response to end input or
enter QUIT to exit.)

bdisk

ARI6188A Enter the output block size of the directory.
(Enter 512 or 4096,
or a null response to use the original size,
or 111(Quit) to exit)

4096

ARI6103A Enter virtual address for new BDISK.
(Enter a null response to end input or
enter QUIT to exit.)

305

ARI6110D Disk 305 is already formatted. Continuing will erase
all data on this disk. Do you want to use the disk?
Enter 0(No), 1(Yes), or 111(Quit).

yes

ARI6146D Are you expanding the SQL/DS™ directory?
Enter 0(No), 1(Yes), or 111(Quit).

no

ARI0647D Do you want to do a CMS FORMAT/RESERVE command on disk 305?
Enter 0(No) or 1(Yes).

yes

ARI6118I Formatting in progress. Please wait...

ARI6131I Copying in progress. Please wait...

ARI6108I Minidisk copied successfully. The SQLDBA SQLFDEF file
will be updated.

ARI6109I SQLDBA SQLFDEF file has been updated on the A disk.

ARI6102A Enter DBEXTENT number (or LOGDSK1, LOGDSK2,
or BDISK) to copy.
(Enter a null response to end input or
enter QUIT to exit.)

ARI0620I SQLDBA SQLFDEF file
successfully copied to production disk.

ARI0673I All COPY DBEXTENT processing completed successfully.

ARI0796I End SQLCDBEX EXEC: 05/23/91 09:09:43 EDT

Ready; T=14.66/24.00 09:09:43

The database manager will now use the new directory disk at address 305. You can confirm this when you start it. For example:

sqlstart db(SQLDBA)

ARI0717I Start SQLSTART EXEC: 05/23/91 16:06:02 EDT.

ARI0320I The default database name is SQLDBA.

ARI0663I FILEDEFS in effect are:

ARISPOOL DISK SPSPEC FILE A1

ARISQLLD DISK TEMSQLLD LOADLIB Q1

ARISQLLD DISK ARISQLLD LOADLIB T1

BDISK DISK 305

:

Using Data Spaces Support with a New Database

If you are creating a new database, you can specify whether it will use Data Spaces Support or the standard DASD I/O system with the directory.

To use Data Spaces Support, you must format the directory disk in 4KB blocks when you create the database. Run the SQLDBINS EXEC and include the parameter:

```
DIRBLK (4096)
```

The SQLDBINS and the SQLDBGEN EXECs are updated in VMDSS to accept this new parameter. For example:

```
sqldbins db(SQLDBA) dirblk(4096)
ARI0717I Start SQLDBINS EXEC: 07/19/91 15:02:24 EDT.
ARI0610D Do you want to install English SQL/DS HELP text?
Enter 0(No), 1(Yes), or 111(Quit).

no
ARI0720I Default DB2 Server for VM bootstrap file SQLDBA SQLRMBT created
on the production disk.
ARI0720I Default DB2 Server for VM bootstrap file SQLDBA SQLDBBT created
on the production disk.
ARI0720I Default DB2 Server for VM bootstrap file SQLDBA SQLISBT created
on the production disk.
ARI0721I Get DB2 Server for VM production minidisk READ access: SQLDBA 195.

ARI0717I Start SQLDBGEN EXEC: 07/19/91 15:02:33 EDT.

ARI0633A Please enter the CUU of the
BDISK disk.

300
ARI0647D Do you want to do a CMS FORMAT/RESERVE command on disk 300?
Enter 0(No) or 1(Yes).

yes
:
```

Note: Make sure that you answer Yes when asked if you want to do a CMS FORMAT/RESERVE (message ARI0647D).

To use the standard I/O system, either include the parameter:

```
DIRBLK (512)
```

or omit the DIRBLK parameter entirely. The default value for DIRBLK is 512.

For more information on the standard SQLDBINS and SQLDBGEN EXEC refer to *DB2 Server for VM System Administration*. For a detailed description of the updated VMDSS version refer to Appendix A, "EXECs" on page 79.

Using Storage Pool Performance Counters

This section describes how to use the DB2 Server for VM counters that reflect the performance of the database for each storage pool, the directory, and unmapped internal dbspaces.

For information on how to interpret these measurements, refer to "Data Spaces Support" on page 64 and "The Standard DASD I/O System" on page 61.

Displaying Pool Counters

VMDSS has several counters that will help you monitor the performance of the DASD I/O systems. To display these counters, use the COUNTER POOL operator command. For a detailed description of this command refer to the *DB2 Server for VSE & VM Operation* manual.

Each storage pool has a set of four counters, which are different depending on whether a particular component is using data spaces. For example, consider a database with two storage pools, pool number 1 uses data spaces while pool number 2 does not:

```
counter pool 1
Counter values at  DATE='05-22-91'  TIME='15:06:33'.

Pool No. 1: Data Spaces
Pages looked at in the buffer  LBUFLOOK: 576
Pages moved from DS to buffer  DSREAD  : 90
Pages moved from buffer to DS  DSWRITE : 0
DS page fault notifications    DSFAULT : 15
ARI0065I Operator command processing is complete.

counter pool 2
Counter values at  DATE='05-22-91'  TIME='15:06:53'

Pool No. 2: *BLOCKIO
Pages looked at in the buffer  LBUFLOOK: 121
Page reads                    PGREAD  : 43
Page writes                   PGWRITE : 0
IUCV *BLOCKIO I/O requests    IUCVBIO : 43
ARI0065I Operator command processing is complete.
```

The directory also has a set of four counters that are different depending on if it is using data spaces. For example, consider two databases. One that is using data spaces with the directory and one that is not:

```
counter pool dir
Counter values at  DATE='05-23-91'  TIME='15:41:07'

Directory: Data Spaces
Pages looked at in the buffer  LBUFLOOK: 71
Pages moved from DS to buffer  DSREAD  : 44
Pages moved from buffer to DS  DSWRITE : 27
DS page fault notifications    DSFAULT : 4
ARI0065I Operator command processing is complete.
```

```
counter pool dir
Counter values at  DATE='05-23-91'  TIME='15:52:23'

Directory: *BLOCKIO
Pages looked at in the buffer  LBUFLOOK: 50
Page reads                    PGREAD  : 23
Page writes                   PGWRITE : 10
IUCV *BLOCKIO I/O requests    IUCVBIO : 33
ARI0065I Operator command processing is complete.
```

Since they can only be used with Data Spaces Support, unmapped internal dbspaces only have one set of counters, For example:

```
counter pool unmapped
Counter values at  DATE='07-29-91'  TIME='17:12:37'

Unmapped Internal Dbspaces:  Data Spaces
Pages looked at in the buffer  LBUFLOOK:  515
Pages moved from DS to buffer  DSREAD  :  21
Pages moved from buffer to DS  DSWRITE :  2
DS page fault notifications    DSFAULT :  0
ARI0065I Operator command processing is complete.
```

If you are using unmapped internal dbspaces, they will not affect the counters for their storage pool. However, mapped internal dbspaces do affect the counters for their pool. Any page movement to do with mapped internal dbspaces will be recorded along with any other page movement in their pool.

Resetting Pool Counters

You can reset the pool counters to zero for any of your storage pools or your directory, by using the RESET POOL command. For example:

```
reset pool 1
Counters reset at  DATE='05-23-91'  TIME='09:26:15'
ARI0065I Operator command processing is complete.
counter pool 1
Counter values at  DATE='05-23-91'  TIME='09:26:19'

Pool No. 1:  Data Spaces
Pages looked at in the buffer  LBUFLOOK:  0
Pages moved from DS to buffer  DSREAD  :  0
Pages moved from buffer to DS  DSWRITE :  0
DS page fault notifications    DSFAULT :  0
ARI0065I Operator command processing is complete.
```

For a detailed description of this command refer to the *DB2 Server for VSE & VM Operation* manual.

Checking the Status of Users

The database manager assigns an “agent” to each user accessing the database, as well as to its own internal tasks, such as checkpoint processing and operator commands. It passes control to each agent in turn, giving each process a slice of its processor time.

If an agent is waiting for a system process to complete (such as a read from DASD) it is referred to as an agent in a *wait state*. The database manager will pass over that agent in favor of another that is not waiting.

There are many types of wait states. The *DB2 Server for VSE & VM Operation* describes the standard states under the SHOW ACTIVE operator command. VMDSS has two additional wait states:

- Data space page fault

- SLD (Save List Definition).

A *data space page fault* occurs when the operating system cannot find a particular data space page in main or expanded storage, and must retrieve it from DASD. A user agent is in a data space page fault wait state when the agent is waiting for the page on DASD to arrive in main storage.

An *SLD (Save List Definition)* block is a control structure used by Data Spaces Support during checkpoint processing. Since there are only four SLD blocks available, the checkpoint agent may need to wait for a free block if they are all being used. You do not need to take any action if you see the checkpoint agent in an SLD wait state.

There are three operator commands that you can use to check whether an agent is in a wait state:

- SHOW ACTIVE
- SHOW LOCK ACTIVE
- SHOW LOCK GRAPH.

If you enter the SHOW ACTIVE operator command, and a user agent is waiting for a data space page fault, you will see:

Agent ... is in DS page fault Wait

For example:

```

show active
Status of SQL/DS agents:
Checkpoint agent is not active
User Agent:  1 User-ID: PETER   is R/W APPL   7B4
Agent is    processing and is in I/O           Wait
User Agent:  2 User-ID: GENE    is R/O SUBS   7B9
Agent is not processing and is in communication Wait
User Agent:  3 User-ID: KAAREL  is R/O APPL   5A4
Agent is    processing and is in DS page fault Wait
User Agent:  4 User-ID: GRANT   is R/W APPL   7BB
Agent is    processing and is in DS page fault Wait
4 agent(s) not connected to an APPL or SUBSYS
ARI0065I Operator command processing complete

```

If the checkpoint agent is waiting for an SLD block, you will see:

Checkpoint agent is in DS SLD block Wait

For example:

```

show active
Status of SQL/DS agents:
Checkpoint agent is in DS SLD block Wait
User Agent:  1 User-ID: PETER   is NEW APPL   106
Agent is waiting for log archive checkpoint.  Wait
User Agent:  2 User-ID: GENE    is NEW SUBS   107
Agent is waiting for log archive checkpoint.
3 agent(s) not connected to an APPL or SUBSYS
ARI0065I Operator command processing complete

```

If you use either the SHOW LOCK ACTIVE or SHOW LOCK GRAPH operator command, you will see DSPF if a user agent is waiting for a data space page fault. For example:

```

show lock active
AGENT  USER      WAIT  TOTAL  LONG  WANTLOCK  WANTLOCK
      USER      STATE LOCKS  LOCKS  TYPE      DBSPACE
1      GILLES    I/O   140    100
2      TUNA      LOCK  226    220   DBSP      12
3      CARSTEN  COMM  97     49
4      EDWIN    LOCK  108    101   DBSP      12
5      ADRIAN   LOCK  107    98    PAGE      14
6      KEVIN    DSPF 88     44
7      KAAREL   DSPF 101    28
ARI0065I Operator command processing complete

```

```

show lock graph kevin
LOCK   LOCK      WAIT  LOCK  DBSP  LOCKK  REQ  REQ
REQUESTER HOLDER  STAT TYPE  NUMBR QUALFIER STATE MODE  DUR
2      KEVIN  4  MARK  DSPF DBSP  15      G WAIT S  LONG
4      MARK  5  MIKE  LOCK DBSP  16      G WAIT X  LONG
5      MIKE  6  LAURA LOCK PAGE  21     88      G WAIT IS LONG
6      LAURA 1  STEVE COMM DBSP  1      G WAIT IX LONG
6      LAURA 3  GRANT DSPF DBSP  1      G WAIT IX LONG
ARI0065I Operator command processing complete

```

For a detailed description of these commands refer to the *DB2 Server for VSE & VM Operation* manual.

Looking at the Buffer Pools

You can check whether a specific local buffer or directory buffer contains a page that comes from a data space by using the SHOW BUFFERS operator command. If a buffer contains a data space page, the **FLAGS** column will contain a hex value where the hex '10' bit (00010000) is turned on. For example:

```

show buffers
Only used buffers are displayed.
DBSPACE  REC      ADDR      FLAGS  FIX CNT
Page Buffers
1         0006FC  0045C000  10    0
32001    000093  0045D000  00     1
1         000080  0045E000  10    0
1         00009D  0045F000  10    0
:
1         00009C  00478000  10    0
1         00009B  00479000  10    0
Directory Buffers
1         000001  0047A000  80     0
1         000002  0047A200  00     0
1         00000D  0047A400  00     0
:
6         000005  0047C800  00     0
31        000001  0047CA00  00     0
ARI0065I Operator command processing is complete.

```

For a detailed description of this command, refer to the *DB2 Server for VSE & VM Operation* manual.

Chapter 4. Measuring Performance

This chapter discusses several ways to measure the performance of an application server that is using VMDSS. It discusses how to collect performance data and convert it into useful measurements.

Understanding Performance Measurements

Performance measurements are relative: they tell how a system behaves for a particular workload. Usually, a system is considered to perform well if it can complete a particular workload faster than other systems or with fewer resources.

In a test system, you can control the workload in your system by running the same tasks many times. During each iteration you can measure how fast your system completed the tasks and how much resource it used.

However, in a production system it is difficult to compare measurements taken at different times because the workload is constantly changing. To obtain a performance measurement, you must compare the average performance of your system measured over a period of time to the workload it processed during that time. To make these comparisons, you need to calculate two types of relative measurements:

- Load
- Effective use

Relative Measurements

Load

Measured as a rate, in tasks per unit time. These measurements help you determine the *load* on the database manager or the operating system over a period of time. High load values in some areas and low load in others may suggest a bottleneck in the system. Also, while similar load measurements do not guarantee that two workloads are comparable, different ones show that the workloads are not comparable.

Effective Use

Measured in a range from 0% to 100% (where 100% indicates optimal performance). These measurements help you estimate how effective the various buffers in the DASD I/O system are performing (remember that data spaces act like a DASD buffer).

Effective use is calculated by comparing the number of pages the system looks for in a buffer to the number it finds there. You can think of this as the number of successes compared to the number of attempts. For example, if the database manager looks in the local buffers for a page 100 times, and finds the page it is looking for 75 times, the local buffers are 75% effective.

This percentage can be calculated in several ways:

$$\frac{\text{Success}}{\text{Attempt}} \times 100\% = \frac{\text{Attempt} - \text{Failure}}{\text{Attempt}} \times 100\% = 1 - \frac{\text{Failure}}{\text{Attempt}} \times 100\%$$

You can also express effective use as a *hit ratio* with the following calculation:

$$\frac{\text{Attempt}}{\text{Failure}}$$

In this case, the higher the value of the hit ratio the better the performance, the lowest value for a hit ratio is 1.

Sampling Interval

An important factor affecting the accuracy of your performance measurements is the sampling interval. Most useful performance values, whether measured directly or calculated from other measures, are averages over time.

If the sampling interval is **too long**, you may lose significant values. For example, you would not see a 10-minute peak in DASD paging load or a 10-minute drop in the effective use of the local buffers if you only looked at the VMDSS performance counters once a day.

If the interval is **too short**, your results may not be statistically valid. For example, if one checkpoint occurred during one 30 minute interval, you could not confidently say that the database manager was performing 2 checkpoints per hour.

You also need to weigh the benefits of making performance measurements against the additional overhead involved. While recording performance numbers every 10 seconds may give you an excellent picture of how your database is working, the additional load on the operating system may reduce your database's performance, or consume large amounts of DASD space.

Measurement Tools

A wide range of tools for monitoring performance is available. This manual describes a set of five that let you measure the performance of the DASD I/O system. Each tool covers a different level of the overall system:

CP Monitor and the VM/Performance Reporting Facility (VM/PRF) product

Measures the performance of the VM/ESA operating system and its resources. Available from IBM. (Refer to page 57.)

CP INDICATE USER and QUERY TIME commands

Measures the performance of your database virtual machine. Included with VM/ESA as a part of CP. (Refer to page 57.)

DATABASE MANAGER COUNTER operator command

Measures the performance of your database. Included with the DB2 Server for VM base product. (Refer to page 59.)

VMDSS SHOW TARGETWS operator command

Measures the amount of main and expanded storage your database machine is currently using. Included with VMDSS. (Refer to *DB2 Server for VSE & VM Operation*.)

VMDSS COUNTER POOL operator command

Measures the performance of individual storage pools within your database and the performance of internal dbspaces and the directory. Included with VMDSS. (Refer to page 61.)

CP Monitor and VM/PRF

The CP Monitor Facility lets you collect data on resource load and effective use. You can control the amount and nature of the data collection, based on the analysis you want to do. To create reports from the collected data, you must either do some programming, or you can use the VM/Performance Reporting Facility (VM/PRF) to produce standard reports. This facility contains four reports helpful in monitoring the overall DASD I/O performance of your database.

CHANNEL_BUSY

Provides a summary of the busiest channel paths ordered by percentage of time the channel is busy. You can use this report to determine the load on your DASD channels, and detect overloaded channels that may benefit from striping.

DASD_BY_ACTIVITY

Provides information on DASD load and effective use, with the most active DASD listed first in the report. You can use this report to detect overloaded DASD volumes that may benefit from:

- Data Spaces Support
- Being moved to less loaded channels

MINIDISK_CACHE_BY_TIME

For processors with expanded storage, provides information on the effective use by CP of minidisk caching, particularly the *read hit ratio*. You can also use this to help determine the best split of expanded storage between minidisk caching and CP system paging.

PAGING_BY_TIME

Provides a summary of the rate of paging between main storage, expanded storage and DASD. You can use this report to determine the load on your CP paging system, and identify cases of excessive paging. Note that paging performed by the standard DASD I/O system (*BLOCKIO interface) is not counted in this report.

USER_RESOURCE_UTIL

Provides information on resource utilization by individual users. Users are listed by CPU consumption. You can use this report to determine the load on the CPU, loads on the paging system, and how many data spaces a particular user owns.

For more information, refer to *VM/PRF User's Guide and Reference*.

CP INDICATE USER and QUERY TIME Commands

These two commands enables you to monitor the overall performance of your database machine. The most important indicators they provide for monitoring VMDSS are:

RES=nnnn

Counts the number of virtual machine pages that are currently in main storage.

READS=nnnnnn

Counts the total number of pages (including data space pages) read from DASD to a virtual machine since it was logged on.

WRITES=nnnnnn

Counts the total number of pages (including data space pages) written from a virtual machine to DASD since it was logged on.

CONNECT=hh:mm:ss

Records the total elapsed time a virtual machine is logged on.

VIRTCPU=mmm:ss

Records the total virtual machine time used since the virtual machine was logged on.

TOTCPU=mmm:ss

Records the total virtual machine time plus the total CP time used (virtual plus overhead) since the virtual machine was logged on.

IO=nnnnnn

Records the total number of I/O requests issued from by the machine since it was logged on. This includes all *BLOCKIO requests. It does not include any pages requested for CP paging.

These commands are only really useful when you use them together. To enter QUERY TIME and INDICATE USER together, type the following from the operator console:

```
#CP QUERY TIME #CP INDICATE USER
```

You also need to compare two consecutive commands. For example, consider the following two QUERY TIME and INDICATE USER commands:

```
#cp query time #cp indicate user
CP QUERY TIME
CP INDICATE USER
TIME IS 15:07:20 EST TUESDAY 01/14/92
CONNECT= 01:21:45 VIRTCPU= 000:06.28 TOTCPU= 000:09.86
USERID=SQLDBA MACH=XC STOR=0009M VIRT=V XSTORE=NONE
IPLSYS=CMS DEVNUM=00031
PAGES: RES=001497 WS=001260 LOCK=000000 RESVD=000000
NPREF=000000 PREF=000000 READS=000130 WRITES=000018
CPU 00: CTIME=01:22 VTIME=000:06 TTIME=000:10 IO=007553
RDR=000000 PRT=000738 PCH=000000
VVECTIME=000:00 TVECTIME=000:00

#cp query time #cp indicate user
CP QUERY TIME
CP INDICATE USER
TIME IS 15:08:31 EST TUESDAY 01/14/92
CONNECT= 01:22:56 VIRTCPU= 000:07.50 TOTCPU= 000:11.82
USERID=SQLDBA MACH=XC STOR=0009M VIRT=V XSTORE=NONE
IPLSYS=CMS DEVNUM=00031
PAGES: RES=001499 WS=001472 LOCK=000000 RESVD=000000
NPREF=000000 PREF=000000 READS=000135 WRITES=000022
CPU 00: CTIME=01:23 VTIME=000:08 TTIME=000:12 IO=009049
RDR=000000 PRT=000974 PCH=000000
VVECTIME=000:00 TVECTIME=000:00
```

The output shows that, during 71 seconds (CONNECT advanced from 01:21:45 to 01:22:56) the following occurred:

- RES** Two additional main storage pages were used (1499-1497)
- READS** Five reads (135-130)

WRITES Four writes (22-18)
VIRTCPU 1.22 seconds of virtual machine time were used (07.50-06.28)
TOTCPU 1.96 seconds of total CPU time were used (11.82-09.86)
IO 1496 I/O requests were issued (9049-7553)

There are four important values that you can calculate from these numbers:

Sampling Interval

CONNECT. The change in elapsed time between CP QUERY TIME commands.

DASD Load

(READS+ WRITES)/sampling interval. Indicates the average load on DASD.

Total CPU Load

(TOTCPU/sampling interval)x100%. Indicates the average proportion of total CPU time your virtual machine is using. While this looks like an effective use measurement, it is really a measure of the load your database machine is placing on the CPU.

I/O Load

IO/sampling interval. Indicates the average load on the I/O system.

For example, from the previous example:

- Sampling Interval: 71 seconds (01:21:45 to 01:22:56)
- DASD Load: 0.127/second (4+5)/71
- Total CPU Load: 2.76% ((1.96/71)x100%)
- I/O Load: 21.07/second (1496/71).

For more information, refer to *VM/ESA: CP Command and Utility Reference*

DB2 Server for VM COUNTER Operator Command

You can use this command as it is documented in the *DB2 Server for VSE & VM Operation* manual. However, there are several important things to remember when you are using Data Spaces Support:

- The database manager does not increment PAGEREAD or PAGWRITE when the operating system retrieves a page from DASD into a data space, writes a page from a data space to DASD, or reads or writes a page to or from the local buffers.
- It does not increment DIRREAD or DIRWRITE when the directory is using Data Spaces Support.

For example, consider a database with two storage pools, each using data spaces, and a directory using the standard DASD I/O system.

After resetting the standard database counters and performing several queries, the COUNTER * command was issued:

```

reset *
Counters reset at DATE='06-06-91' TIME='13:27:00'
ARI0065I Operator command processing is complete.
counter *
Counter values at DATE='06-06-91' TIME='13:58:12'
Calls to RDS RDSCALL : 68
Calls to DBSS DBSSCALL: 139
LUWs started BEGINLUW: 58
LUWs rolled back ROLLBACK: 11
System checkpoints taken CHKPOINT: 1
Maximum locks exceeded LOCKLMT : 0
Lock escalations ESCALATE: 0
Waits for lock WAITLOCK: 0
Deadlocks detected DEADLCK : 0
Looks in page buffer LPAGBUFF: 298722
DBSPACE page reads PAGEREAD: 0
DBSPACE page writes PAGWRITE: 0
Looks in directory buffer LDIRBUFF: 5054
Directory block reads DIRREAD : 4014
Directory block writes DIRWRITE: 120
Log page reads LOGREAD : 2
Log page writes LOGWRITE: 40
Total DASD reads DASDREAD: 4524
Total DASD writes DASDWIT: 49
Total DASD I/O DASDIO : 3986
Lock timeouts detected LTIMEOUT: 0
ARI0065I Operator command processing is complete.

```

While there were 298722 looks into the local page buffers, the database manager did not record any reads or writes to DASD for dbspaces. It did record 4014 reads and 120 writes, but they were for the directory. To see how many times the operating system accessed DASD you need to use the COUNTER POOL command. Refer to “Data Spaces Support” on page 64.

There are five important values that you can calculate from the COUNTER command:

Sampling Interval

TIME. The elapsed time between the RESET and the COUNTER command.

LUW Load

BEGINLUW/sampling Interval. This is the average rate at which the database manager receives logical units of work. It measures the average load on the database machine.

Checkpoint Load

CHKPOINT/sampling interval. This is the average rate of checkpoints. Checkpoints are overhead; they represent an additional load on the database machine.

Local Buffers Effective Use (Only When not using Data Spaces Support)

$(1 - \text{PAGEREAD} / \text{LPAGBUFF}) \times 100\%$. Indicates the percentage of time the database manager found a page in the local buffers and did not need to retrieve it from DASD. Ranges from 0 to 100%, where 100% means that every page the database manager needed was in the local buffers.

This information can also be expressed as the *local buffers hit ratio* ($\text{LPAGEBUFF} / \text{PAGEREAD}$).

Directory Buffers Effective Use (Only When not using Data Spaces Support)

($1 - \text{DIRREAD} / \text{LDIRBUFF}$)x100%. Indicates the percentage of time the database manager found a page in the directory buffers and did not need to retrieve it from DASD. Ranges from 0 to 100%, where 100% means that every directory page the database manager needed was in the directory buffers.

This information can also be expressed as the *directory buffers hit ratio* ($\text{LDIRBUFF} / \text{DIRREAD}$).

For example, from the previous output:

- Sampling Interval: 1872 seconds (13:58:12-13:27:00)
- LUW Load: 3.54/second (58/1872)
- Checkpoint Load: 2/hour (1/1872) This value is not statistically valid. You need to monitor checkpoints over a longer period of time for an accurate calculation.
- Local Buffers Effective Use: Not a valid calculation because at least one storage pool is using Data Spaces Support.
- Local Buffers Hit Ratio: Not valid.
- Directory Buffers Effective Use: 20.58% ($1 - 4014 / 5054$)X100% (Note, this measurement is still valid because the directory is not using Data Spaces Support.)
- Directory Buffers Hit Ratio: 1.26 (5054/4014).

VMDSS COUNTER POOL Operator Command

VMDSS has several counters to help you monitor the performance of the standard DASD I/O system and of VM Data Spaces Support. Each storage pool has two sets of counters. The system increments and displays one set for pools using Data Spaces Support, and the other for pools using the standard DASD I/O system. (These two sets of counters are also supported for the directory and separate unmapped internal dbspaces.)

The Standard DASD I/O System

The set of counters for the standard I/O system includes:

LBUFLOOK

The number of times the database manager has looked into the local data page buffers for a page from this counter's storage pool.

PGREAD

The number of pages in the pool moved from DASD to the local buffers.

PGWRITE

The number of pages in the pool moved from the local buffers to DASD.

IUCVBIO

The number of IUCV *BLOCKIO requests submitted by the database manager to the operating system for this pool. Note that, because the database manager often uses one IUCV *BLOCKIO request to write several pages at once, IUCVBIO is usually lower than PGREAD+PGWRITE.

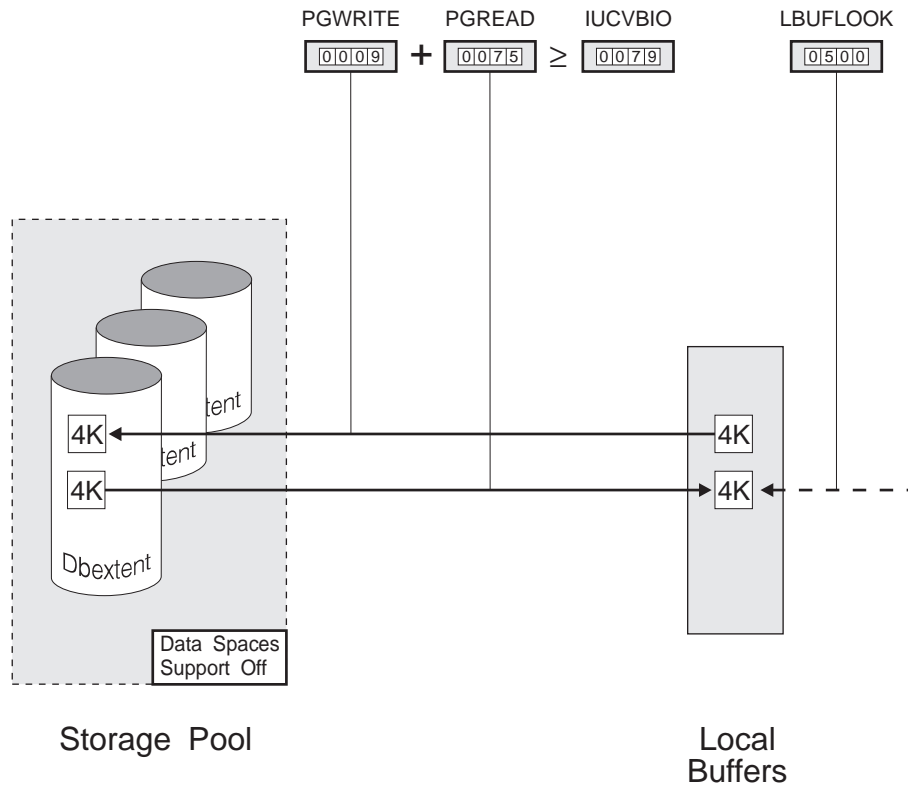


Figure 15. Pool Counters Using the Standard DASD I/O System

For example:

```

reset pool 1
Counters reset at DATE='07-23-91' TIME='11:06:02'
ARI0065I Operator command processing is complete.

counter pool 1
Counter values at DATE='07-23-91' TIME='11:42:12'

Pool No. 1: *BLOCKIO
Pages looked at in the buffer LBUFLOOK: 255311
Page reads PGREAD : 172070
Page writes PGWRITE : 67
IUCV *BLOCKIO I/O requests IUCVBIO : 172125
    
```

The output shows that during 2170 seconds (from 11:06:02 to 11:42:12) the following occurred for storage pool 1:

LBUFLOOK

The database manager looked at the local page buffers 255,311 times

PGREAD

It read 172,070 pages from DASD

PGWRITE

It wrote 67 pages to DASD

IUCVBIO

It submitted 172,125 IUCV *BLOCKIO requests to the operating system.

There are six important values that you can calculate from the standard DASD I/O counters:

Sampling Interval

TIME. The elapsed time between the RESET POOL and the COUNTER POOL command.

Standard Local Buffer Looks Load

LBUFLOOK/sampling interval. The average rate of looks into the local buffers for this pool. It indicates the load that this storage pool places on the database machine. It may be useful to compare this value across storage pools and to LPAGBUFF.

Standard DASD Load

IUCVBIO/sampling interval. The average rate of IUCV *BLOCKIO requests for this pool. It represents the load the database manager places on the operating system's IUCV *BLOCKIO DASD system.

Read Percentage

$PGREAD/(PGWRITE+PGREAD) \times 100\%$. The percentage of DASD accesses that were reads. 100% indicates that every time the database manager accessed DASD, it was to read a page.

This information can also be expressed as the *read-to-write ratio* ($PGREAD/PGWRITE$).

Local Buffers Effective Use

$(1 - PGREAD/LBUFLOOK) \times 100\%$. The percentage of times that the database manager found a page in the local buffers. This represents how effective the local buffer pool is. 100% indicates that every page needed by the database manager was in the local buffer pool.

This information can also be expressed as the *local buffers hit ratio* ($LBUFLOOK/PGREAD$).

Standard Blocking Effective Use

$(1 - (IUCVBIO - PGREAD)/PGWRITE) \times 100\%$. The percentage of write requests that the database manager blocked together. This represents how effective the database manager is at using one IUCV *BLOCKIO request to write several pages.

This information can also be expressed as the *standard blocking hit ratio* ($PGWRITE/(IUCVBIO - PGREAD)$).

For example, from the previous output:

- LBUFLOOK=255311
- PGREAD=172070
- PGWRITE=67
- IUCVBIO=172125
- Sampling Interval: 2170 seconds (11:42:12-11:06:02)
- Standard Buffer Looks Load: 117.65/second (255311/2170)
- Standard DASD Load: 79.32/second (172125/2170)

- Read Percentage: 99.96% ($172070/(67+172070)$)x100%
- Read-to-Write Ratio: 2568.2 (172070/67)
- Local Buffer Effective Use: 32.6% ($1-(172070/255311)$)x100%
- Local Buffer Hit Ratio: 1.48 (255311/172070)
- Standard Blocking Effective Use: 17.91% ($1-(172125-172070)/67$)x100%.

While this example described the standard storage pool counters for a single pool, you can analyze the counters for the directory in the same way.

Data Spaces Support

The set of counters for storage pools using VM Data Spaces support includes:

LBUFLOOK

The number of times the database manager has looked into the local data page buffers for a page from this counter's storage pool.

DSREAD

The number of pages in the pool moved from data spaces to the local buffers.

DSWRITE

The number of pages in the pool moved from the local buffers to data spaces.

DSFAULT

The number of pages in the pool requested from data spaces, that the operating system did not find (faults) in main or expanded storage.

A page will be in main or expanded storage if it was:

- Fetched on a previous occasion and is still in storage (refer to "Reading Pages" on page 4).
- Part of a block of pages that were prefetched (refer to "Blocking and Prefetching" on page 70).

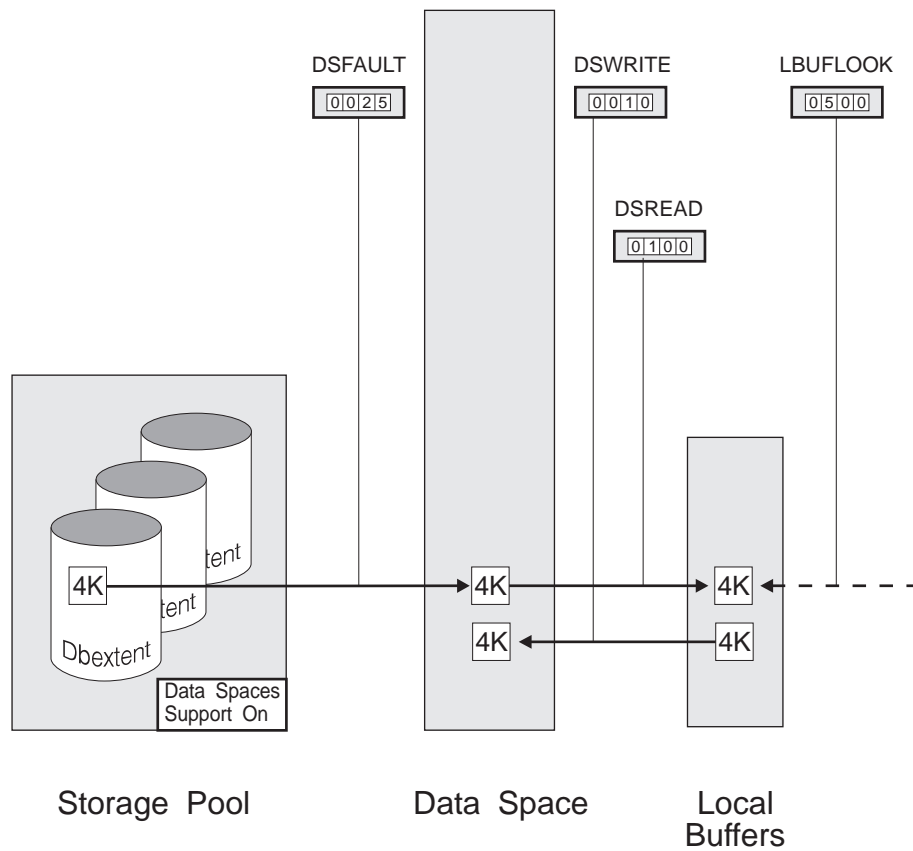


Figure 16. Pool counters used with Data Spaces Support

For example:

```

reset pool 2
Counters reset at DATE='07-23-91' TIME='11:04:49'
ARI0065I Operator command processing is complete.

counter pool 2
Counter values at DATE='07-23-91' TIME='11:37:11'

Pool No. 2: Data Spaces
Pages looked at in the buffer LBUFLOOK: 310615
Pages moved from DS to buffer DSREAD : 222322
Pages moved from buffer to DS DSWRITE : 32
DS page fault notifications DSFAULT : 50977
ARI0065I Operator command processing is complete.

```

The output shows that during 1942 seconds (from 11:04:49 to 11:37:11) the following occurred for storage pool 2:

- The database manager looked for storage pool 2 pages in the local page buffers 310615 times
- It read 222322 pages from data spaces

- It wrote 32 pages to data spaces
- The operating system could not find 50977 pages in main or expanded storage and had to access DASD.

There are eight important values that you can calculate from the Data Spaces Support counters:

Sampling Interval

TIME. The elapsed time between the RESET POOL and the COUNTER POOL command.

Local Buffer Looks Load

LBUFLOOK/sampling interval. The average rate of looks into the local buffers for this pool. It is an indicator of the relative load that this storage pool places on the database machine. It may be useful to compare this value across storage pools.

Data Spaces Paging System Load

(DSREAD+DSWRITE)/sampling interval. The average rate that the database manager accesses data spaces for this pool. This represents the load the database manager places on the Data Spaces Support system.

Data Spaces DASD Read Load (Approximate)

DSFAULT/sampling interval. The average rate that the operating system accesses DASD for this pool. This represents the load the Data Spaces Support system places on DASD.

Data Spaces Read Percentage

$DSREAD/(DSWRITE+DSREAD) \times 100\%$. The percentage of data space accesses that were reads. 100% indicates that every time the database manager accessed a data space, it was to read a page.

This information can also be expressed as the *data spaces read-to-write ratio* (PGREAD/PGWRITE).

Local Buffers Effective Use

$(1 - DSFAULT/LBUFLOOK) \times 100\%$. The percentage of times that the database manager found a page in the local buffers. This represents how effective the local buffer pool is. 100% indicates that every page needed by the database manager was in the local buffer pool.

This information can also be expressed as the *local buffers hit ratio* (LBUFLOOK/DSREAD).

Data Spaces Effective Use (Approximate)

$(1 - DSFAULT/DSREAD) \times 100\%$. The percentage of times that the database manager found a page in main or expanded storage that the database manager could not find in the local buffers. This represents how effective the Data Spaces Support paging system is. 100% indicates that every page that the database manager could not find in the local buffers was in main or expanded storage.

This information can also be expressed as the *data spaces hit ratio* (DSREAD/DSFAULT).

Overall VMDSS Effective Use

$(1 - DSFAULT/LBUFLOOK) \times 100\%$. The percentage of times that a page needed by the database manager was in main or expanded storage. This represents how effective the local buffers and the Data Spaces

Support paging system are together. 100% indicates that every page needed by the database manager was in main or expanded storage.

This information can also be expressed as the *overall VMDSS hit ratio* (LBUFLOOK/DSFAULT).

For example, from the previous output:

- LBUFLOOK=310615
- DSREAD=222322
- DSWRITE=32
- DSFAULT=50977
- Sampling Interval: 1942 seconds (11:37:11-11:04:49)
- Local Buffer Looks Load: 159.95/second (310615/1942)
- Data Spaces Paging System Load: 114.5 ((222322+32)/1942)
- Data Spaces DASD Read Load: 26.12/second (50977/1952)
- Data Spaces Read Percentage: 99.99% (222322/(222322+32))
- Data Spaces Read-to-Write Ratio: 6947.6 (222322/32)
- Local Buffers Effective Use: 28.43% (1-(222322/310615))
- Local Buffers Hit Ratio: 1.40 (310615/222322)
- Data Spaces Effective Use: 77.07% (1-(50977/222322))
- Data Spaces Hit Ratio: 4.36 (222322/50977)
- Overall VMDSS Effective Use: 83.59% (1-(50977/310615))
- Overall VMDSS Hit Ratio: 6.09 (310615/50977)

While this example described the data space counters for a single storage pool, you can analyze the counters for the directory and unmapped internal dbspaces in the same way.

Chapter 5. Tuning Performance

This chapter describes the various configuration options and tuning parameters that you can use to optimize the performance of your application server with VMDSS.

Deciding When to Use Data Spaces

This section describes the advantages of using Data Spaces Support over the standard DASD I/O system, and when to use Data Spaces Support with:

- Storage Pools
- Internal Dbspaces
- The Directory.

Advantages

The paging system in Data Spaces Support can be much faster and more efficient than the standard DASD I/O system.

The data spaces act like a large DASD cache, keeping the most recently used data in the fastest storage. While this is similar to using a large pool of buffers or DASD caching, there are significant advantages to using Data Spaces Support over these two methods. (Refer to *DB2 Server for VM System Administration* for more information on buffer pools.)

Some of the advantages are:

- Shorter path length
- Asynchronous page fault processing
- Striping
- Blocking and prefetching
- Dynamic working storage size management
- More asynchronous writes.

These are described in turn below.

Shorter Path Length

There is a series of internal processes between when the database manager requests a page from DASD, and when the operating system transfers it to main storage. This series is shorter when you use Data Spaces Support than when you use the standard DASD I/O system.

Asynchronous Page Fault Processing

Since the operating system treats the buffers like part of the database manager code, it may page them out to system paging DASD if it needs main storage. Whenever the database manager needs a piece of code (or a buffer) that has been moved to paging DASD, it and all its users must wait for that page to return from DASD.

With Data Spaces Support, you can use a smaller pool of local buffers, decreasing the chance of a buffer being paged out. If a page fault occurs in a data space (the operating system cannot find the page in main or expanded storage) the database

manager can proceed with other users and return to the original user when the fault has been resolved.

Striping

When you use striping, the database system tries to keep related data pages physically close together on DASD. (It allocates pages in groups of 16.) Thus, when the operating system needs to retrieve related pages from DASD, there is a good chance that the pages will be located together. The operating system can then read in a whole series of pages with one I/O operation, which improves the performance of your system.

Striping also spreads these groups of 16 pages across all the dbextents in a storage pool. If the dbextents are on separate physical devices, the operating system can read several groups of pages at the same time (asynchronously). This improves blocking and prefetching (see below), and helps you balance the load between DASD packs.

Blocking and Prefetching

When you use the Data Spaces Support, the operating system tracks the way you access pages. It records which pages you have used together (in a *block*) and the order in which you use them. Then, when the database manager requests a page from a data space, if the page is on DASD, CP will start retrieving (*prefetching*) other pages in the same block in the order you previously followed. Since DASD I/O can proceed in parallel (because of striping), this effectively places pages in main storage before the database manager needs them.

In some cases, the database manager will pass information to the operating system about how it expects to use pages. The operating system uses this information to modify its own reference pattern and thereby further improve prefetching.

Dynamic Working Storage Size Management

You can dynamically manage how the database manager uses main and expanded storage:

- You can set a target working storage size (refer to “Target Working Storage Size Parameter” on page 12) to control how much main and expanded storage your database machine uses.
- You can favor some storage pools over others by setting their working storage residence priority (refer to “Working Storage Residency Priorities” on page 12). This lets you improve the performance of critical storage pools, even if you have a limited amount of main and expanded storage.
- You can set a save interval (refer to “The Save Interval” on page 12). When the number of blocks of modified pages in a data space exceeds this parameter, the database manager directs the operating system to write all the modified pages in that data space to DASD. This reduces the number of modified pages in storage. As a result, there are fewer pages to be saved during checkpoint processing, which reduces checkpoint processing time.

More Asynchronous Writes

With Data Spaces Support, the database manager can write modified pages back to DASD (refer to “Modifying Pages” on page 7) “more” asynchronously than without it.

With Data Spaces Support off: If the database manager needs a buffer occupied by a modified page, it first writes the page to DASD, then loads the buffer with a new page.

When it does this, it puts the current agent (refer to “Checking the Status of Users” on page 50) into an I/O Wait State until the write is complete. Since the database manager continues to service agents that are not in wait states, this process is asynchronous *between agents*.

With Data Spaces Support on: When the database manager writes a modified page to a data space, the current agent is not put into a wait state. The operating system ensures that the page is eventually written to DASD (before the next checkpoint) without stopping the current agent. This process is asynchronous *within an agent* and therefore more asynchronous than without Data Spaces Support.

Storage Pool

We suggest that you turn Data Spaces Support on for all your storage pools. Even without adding main or expanded storage to facilitate caching, you should see performance improve due to the advantages of shorter path length, striping, blocking, and prefetching.

If you want the additional benefit of caching, you should first consider the cost in main and expanded storage. Whenever you use Data Spaces Support, the operating system will use main and expanded storage to cache any data the database manager uses. If the database manager needs this data again, the operating system can retrieve it quickly. However, if the cached data is not used very often, it may be swapped out of main or expanded storage before it is referenced again. If this happens, you are using main or expanded storage to cache pages without receiving any of the benefit. Thus, if the main and expanded storage in your system is limited, you should only use caching for your most active pools.

(You can effectively turn caching off for a particular storage pool without turning Data Spaces Support off, by using working storage residence priority “1.” Refer to “Choosing Storage Residence Priorities” on page 73 .)

Internal Dbspaces

You can improve the performance of your database by using unmapped internal dbspaces. We suggest that you do so unless you do not have enough VM paging DASD (refer to “VM/ESA Paging DASD” on page 19).

Unmapped internal dbspaces have the following advantages over mapped ones:

- You can use all the space in your storage pools for public and private dbspaces.
- The database manager never writes unmapped internal dspace pages to DASD. This reduces your overall DASD I/O, without affecting the integrity of your system. (You do not need a record of the internal dbspaces to recover

your database.) Note that the operating system may still swap unmapped internal dbspace pages to VM paging DASD.

If you want to manage your internal dbspaces the same way you manage all your other dbspaces, you may want to use mapped internal dbspaces. If you place your internal dbspaces in a separate storage pool, you can turn Data Spaces Support on or off, and set a working storage residence priority for them.

For information on how to customize your database for internal dbspaces, refer to “Using Data Spaces with Internal Dbspaces” on page 44.

Directory

We suggest that you use Data Spaces Support with the directory. However, you may choose not to if you need to switch your database machine between XC mode and ESA mode.

Every time you switch to a processor or operating system that does not support XC mode (for example a backup system) you must reblock the directory disk. (Refer to “Reblocking the Database Directory” on page 45.)

For information on how to customize a database to use Data Spaces Support with the directory, refer to “Using Data Spaces with the Directory” on page 45.

Managing Your Working Storage Size

Working storage is composed of:

- The database manager code and the storage it uses to hold control information (control blocks)
- The directory buffers
- The local buffers
- Data space pages in main and expanded storage, including those in public, private and internal dbspaces.

While you do not have direct control over how much storage the database manager and its control blocks use (refer to *DB2 Server for VM System Administration* under “virtual storage requirements”), you can control the amount of storage used by the directory buffers, the local buffers, and by data space pages.

The amount of storage used by the directory buffers is $NDIRBUF * 560$ bytes, where $NDIRBUF$ is the number of directory buffers. This applies whether you are using Data Spaces Support or not. The storage used by the local buffers is $NPAGBUF * 4144$ bytes. (Each buffer page requires 48 bytes of overhead. For example a 4KB page requires $4096 + 48$ bytes or 4144 bytes of storage.) By reducing or increasing the number of directory and local buffers you are using you can reduce or increase your working storage.

There are five parameters to help you manage the number of data space pages in main and expanded storage that your database machine uses.

- Target working storage size
- Working storage residency priority
- `SEPINTDB` (mapped or unmapped internal dbspaces)
- Checkpoint interval

- Save interval.

These parameters are discussed in the following sections.

Choosing the Target Working Storage Size

The target working storage parameter (TARGETWS) helps you to balance the amount of main and expanded storage used by your database machine, with the amount used by other virtual machines in your VM system.

If you set TARGETWS **too low**, you may unnecessarily restrict the amount of available storage your database machine can use. You may also find that the operating system does not release pages fast enough and your current storage size always exceeds your target. If this happens, some working storage residence priorities are not effective. Remember, the database manager starts releasing most pages when the target working storage size is reached: if your current storage size is always greater than your target, the database manager only keeps those pages with a residence priority of 4 or 5. In this case, pages with any other priority will not be differentiated.

If you set it **too high**, your database machine may never reach the target you set. VM may not give your database machine the amount of main and expanded storage it asks for. You may find that the operating system restricts your working storage size before the database manager does. If this happens, the database manager only releases those pages with a residence priority of 1 or 2.

Once you find an acceptable target working storage size, it is important not to let your current size exceed it by too much. If it does it can have the same effect as setting TARGETWS too high (VM restricts storage). If your working storage is too high, it means that you are either:

- Keeping too many modified pages in main or expanded storage (reduce SAVEINTV)
- Using too many unmapped internal dbspace pages (use mapped internal dbspaces instead)
- Setting your storage residence priorities too high (use a setting of 3 or less).

For information on managing modified pages, refer to “Managing Checkpoints” on page 75, and for information on unmapped pages refer to “Unmapped Internal Dbspaces” on page 74. For more information about how the TARGETWS mechanism works, refer to Appendix F, “Why is the TARGETWS Value Frequently Exceeded?” on page 101.

Choosing Storage Residence Priorities

If you set a realistic target working storage size (large enough to be effective but not so large as to overload the operating system), you will be able to use storage residence priorities to favor pages from certain storage pools. When the database manager copies a page from a data space into its buffers, it checks the residence priority of that page. At the default value of 3, it releases the data space page from main and expanded storage if the current working storage size is greater than the target. However, the buffer page stays in the buffer pool until the database manager needs the space for a new page.

For most applications, the default priority should be correct. However, if you can identify certain storage pools as “high priority” pools that contain

performance-critical dbspaces, you can favor them by assigning them a high residence priority. Low priority pools can be assigned a low residence priority.

You can assign one of five storage residence priorities:

- 1 The database manager always releases pages when possible, regardless of the current working storage size. This effectively turns caching off. (For low re-used pages.)
- 2 The database manager always releases pages, except index pages, when possible. It will only release index pages when the current working storage size exceeds your target. (For low re-used pages, randomly accessed using indexes.)
- 3 The database manager releases pages when the current working storage size exceeds your target. This is the default priority.
- 4 The database manager releases data pages when the current working storage size exceeds your target. It does not release index pages. (For high re-used pages, randomly accessed using indexes.)
- 5 The database manager never releases pages. (For only the most re-used or most important pages where dbscans are frequent.)

Table 3 summarizes the five storage residence priorities. An R indicates that the database manager releases a page from main and expanded storage after it has been moved to a local buffer.

Table 3. Storage Residence Priorities

Page Type	Current Working Storage Size	Working Storage Residence Priority				
		1	2	3	4	5
Data	≤ target	R	R			
	> target	R	R	R	R	
Index	≤ target	R				
	> target	R	R	R		

Unmapped Internal Dbspaces

Whether you are using mapped or unmapped internal dbspaces also affects your current working storage size. (The operating system controls it by moving pages from main storage to and from system paging DASD.) However, unmapped pages are included in your total current working storage size, and can inflate it beyond your target.

For example, if you are performing operations that use large amounts of internal dspace storage (creating large indexes, or sorting large tables), you may fill unmapped internal dbspaces with pages that are not released until the index or sort is complete. Even if this increases your current working storage above your target, the database manager will not release internal dspace pages internal dspace pages when it no longer needs the internal dspace; frequently not until the end of a logical unit of work.

Managing Checkpoints

A checkpoint is an internal operation where the database manager writes modified data and status information to DASD, and writes a summary status record to the log.

When the database manager takes a checkpoint:

- It writes all modified data and directory pages back to DASD.
- It frees all *shadow pages*. (Whenever it “modifies” a page in a storage pool, it creates a new page in the same pool, and keeps the original as a shadow page.)
- It writes the log buffer out to the log disks.
- If LOGMODE=Y (no archive), the database manager clears space in the log up to the beginning of the oldest LUW still active when the checkpoint is taken.
- It updates the directory pages to account for released shadow pages and updated page allocation maps.

A checkpoint has two performance implications:

- It performs a high amount of I/O to DASD. It writes all the modified buffer pages and data space pages back to DASD, and updates the directory disk.
- It holds up processing. User agents must wait until the checkpoint is finished before they can proceed.

Choosing the Checkpoint Interval

To control the duration between database checkpoints, use the CHKINTVL initialization parameter. This parameter specifies how many log pages the database manager will fill before it takes its next checkpoint.

Setting the Time Between Checkpoints

The time between checkpoints depends on the number of modifications you make to the database. If logging is turned on, the database manager writes to the log every time you perform an insert, update, or delete. The more modifications you make, the faster you will reach a checkpoint. If you only perform queries, the database manager may never perform a checkpoint.

We recommend that you adjust the CHKINTVL parameter so that the database manager takes a checkpoint every 10 to 15 minutes. Should you experience a system error, it should take you no longer than 10 to 15 minutes to restart the database manager after you have recovered your system. If you adjust CHKINTVL so that checkpoints occur less frequently, for example every four hours, it may take up to or more than four hours to restart your database.

If you set the CHKINTVL parameter **too low**, you minimize the risk of filling the log or storage pools. However, while each checkpoint is faster, you increase the overall number of checkpoints.

If you set it **too high**, you lower the overhead associated with checkpoint processing. However, you risk filling the log and storage pools, and you increase the time required to complete a checkpoint. It may also take longer to recover from a system error.

Choosing the Save Interval

The SAVEINTV parameter limits the number of modified pages in main and expanded storage. When the number of blocks of modified pages in a data space exceeds this parameter, the database manager directs the operating system to write all the modified pages in that data space to DASD.

This is done asynchronously, meaning that the database manager can continue servicing other users while the save completes.

If you set the save interval appropriately, you can reduce the time it takes to perform a checkpoint. While the checkpoints will take place at the same intervals (the database manager still fills log pages at the same rate), they will be shorter because there will be fewer modified pages to write to DASD.

While the default setting should work well for most databases, you may consider changing it. If you find that your checkpoints take too long, reduce SAVEINTV. If checkpoint processing is not a problem, consider increasing it.

You may also need to reduce SAVEINTV if your current working storage size is always much larger than your target. The database manager does not release modified pages from main storage until a save interval or a checkpoint. So if you are using a high SAVEINTV, and performing many inserts, updates, or deletes, the database machine may keep too many modified pages in main storage.

You can compare the number of times the database manager requests the operating system to save pages to the number of times it performs a checkpoint, by using the COUNTER and COUNTER INTERNAL operator commands. The COUNTER command displays the CHKPOINT counter, which records the number of checkpoints that occurred since the last time you reset the counter. The COUNTER INTERNAL command displays the SAVEGNRL counter. SAVEGNRL counts the number of times the database manager directs the operating system to write all the modified pages in a data space to DASD. If you reset both the CHKPOINT and SAVEGNRL counters at the same time, you can monitor the number of save requests between each checkpoint.

For more information on the COUNTER operator command, refer to the *DB2 Server for VSE & VM Operation* manual. For more information on the COUNTER INTERNAL command, refer to Appendix E, “Internal Counters” on page 97.

Using Striping

Striping evenly distributes all new and modified pages across all the dbextents in a storage pool. We suggest that you use striping, even if you are not using Data Spaces Support.

For information on how to use striping, refer to “Turning Striping On and Off” on page 41.

With One Dbextent Per Pool

You may choose not to use striping for a particular storage pool if it has only one dbextent, because in that case the database manager cannot distribute your data across several dbextents in the storage pool.

However, even with only one dbextent you may find a small performance improvement. The database manager still allocates space on the disk in blocks of 16 4KB-pages. By doing this it improves the probability that the pages you need are close together.

One Dbextent Per Device: For the storage pools that will use striping, it is recommended that you assign each dbextent in the pool to a separate physical storage device. While the database manager distributes pages across dbextents, it does not recognize whether those dbextents are on the same physical device or several different ones. If you assign two dbextents to one physical storage device, performance will be degraded, because the database manager cannot retrieve pages from both dbextents in parallel.

Dbextent Size: If you plan to use striping, you should define several dbextents of the same size in each storage pool. If you have large and small dbextents mixed in the same pool, you may find that the database manager does not distribute pages evenly across them. Rather, it distributes pages across all the dbextents until the smallest one is full. It then continues to fill the larger dbextents.

Number of Dbextents: For best performance, use at least four dbextents per storage pool. CP will only prefetch pages from four dbextents in a storage pool simultaneously. Any less than four means that CP does not have as many devices as possible to prefetch from in parallel. (Refer to “Blocking and Prefetching” on page 70 for more information on prefetching.)

Using Striping with Existing Data: Striping only evenly distributes new or modified pages. It does not reallocate existing pages. To ensure that striping works with all your pages, unload all the dbspaces in your database, and reload them with striping turned on. This makes all the pages “new pages.”

Choosing Logical or Physical Mapping

Logical mapping is the default and recommended type of mapping for most applications. However, applications that perform mostly updates **may** perform better with physical mapping.

Because you can only change the mapping parameter at startup time, you should always use logical mapping for your production applications, and consider physical for single-user-mode dataloads.

Real Storage Requirements for Data Spaces

For each data space which is larger than 1024 megabytes, CP must keep two contiguous real storage pages until the database is shut down. These two pages are required for CP segment tables and must remain in real storage at all times. If you are using VMDSS with many databases or with a very large database, and have a constrained real storage environment, this will further reduce any real storage availability and increase system paging.

The only way to increase real storage availability in these situations is to reduce the number of databases using data spaces, or reduce the number of storage pools which are mapped to data spaces, or both.

For each data space which is less than or equal to 1024 megabytes, CP must keep one real storage page until the database is shut down.

Appendix A. EXECs

The following table lists the CMS EXECs updated for VMDSS. As well as a brief description of each EXEC, the table contains page references to other sections of the manual where you can find out more about each one. The page numbers under **Reference**, point to detailed descriptions of the changes to each EXEC.

Operating	Reference	EXEC	Function
45	79	SQLCDBEX	Copies, expands, or reblocks the directory disk.
48	80	SQLDBGEN	Generates a new database with a directory disk formatted in either 512B or 4KB blocks.
48	82	SQLDBINS	Generates a new database by calling SQLDBGEN, and loads the DBS Utility, ISQL, sample tables, help text, and the FIPS flagger.

SQLCDBEX

Copies, expands, or reblocks the database directory.

Syntax

```
▶▶—SQLCDBEX—Dbname(server_name)————▶▶
```

Description

This EXEC is updated for VMDSS. It now asks you which block size you want the output directory to be. For a block size of 512 bytes, type 512. For a block size of 4KB, type 4096. If you do not type in a size and just press Enter, the EXEC will make the output directory the same block size as the input directory.

Note: Refer to *DB2 Server for VM System Administration* for a complete description of this EXEC and all its parameters.

Example

Consider a database with a B-disk (directory disk) at address 300 and a block size of 512 bytes. To reblock the directory disk to 4096 (4K) bytes, you need to run the SQLCDBEX EXEC to copy the directory onto a new disk (305) blocked to 4K. For example:

```

sqlcldbex db(SQLDBA)
ARI0717I Start SQLCDBEX EXEC: 05/23/91 08:58:36 EDT.
ARI6102A Enter DBEXTENT number (or LOGDSK1, LOGDSK2,
or BDISK) to copy.
(Enter a null response to end input or
enter QUIT to exit.)

bdisk
ARI6188A Enter the output block size of the directory.
(Enter 512 or 4096,
or a null response to use the original size,
or 111(Quit) to exit)

4096
ARI6103A Enter virtual address for new BDISK.
(Enter a null response to end input or
enter QUIT to exit.)

305
ARI6110D Disk 305 is already formatted. Continuing will erase
all data on this disk. Do you want to use the disk?
Enter 0(No), 1(Yes), or 111(Quit).

yes
ARI6146D Are you expanding the SQL/DS directory?
Enter 0(No), 1(Yes), or 111(Quit).

no
ARI0647D Do you want to do a CMS FORMAT/RESERVE command on disk 305?
Enter 0(No) or 1(Yes).

yes
ARI6118I Formatting in progress. Please wait...
ARI6131I Copying in progress. Please wait...
ARI6108I Minidisk copied successfully. The SQLDBA SQLFDEF file
will be updated.
ARI6109I SQLDBA SQLFDEF file has been updated on the A disk.

ARI6102A Enter DBEXTENT number (or LOGDSK1, LOGDSK2,
or BDISK) to copy.
(Enter a null response to end input or
enter QUIT to exit.)

ARI0620I SQLDBA SQLFDEF file
successfully copied to production disk.
ARI0673I All COPY DBEXTENT processing completed successfully.
ARI0796I End SQLCDBEX EXEC: 05/23/91 09:09:43 EDT
Ready; T=14.66/24.00 09:09:43

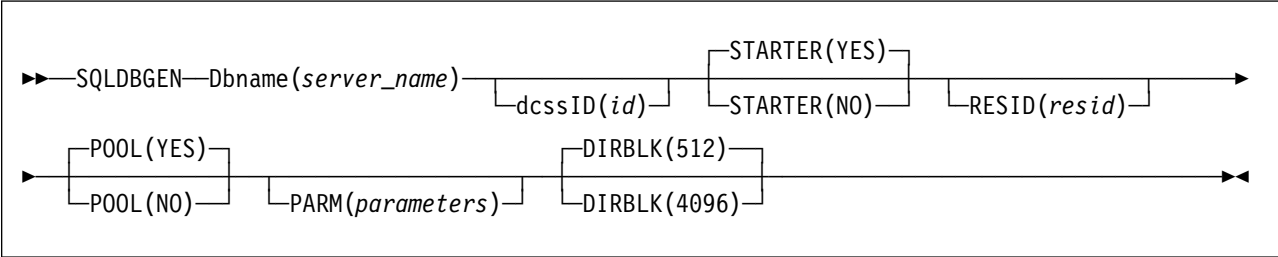
```

The database manager will now use the new directory disk at address 305.

SQLDBGEN

Generates a new database with a directory disk formatted in either 512B or 4KB blocks.

Syntax



DIRBLK

Specifies whether the SQLDBGEN EXEC will generate the new directory in 512 or 4KB blocks. The default is 512. This parameter is only effective if you answer YES to message ARI0647D, which asks whether you want to format the directory disk.

Note: Refer to *DB2 Server for VM System Administration* for a complete description of this EXEC and all its parameters.

Description

This EXEC is updated for VMDSS, and now includes a new parameter **DIRBLK**. You can use this EXEC as described in *DB2 Server for VM System Administration*. Just make sure that if you want to use Data Spaces Support with the directory of your new database, you include the parameter DIRBLK (4096).

Example

To generate a new database with a directory disk at address 300 blocked to 4KB, run the SQLDBGEN EXEC with parameter DIRBLK(4096). For example,

```

sqldbgen db(SQLDBA) dirblk(4096)
ARI0717I Start SQLDBGEN EXEC: 07/19/91 15:19:43 EDT.
ARI0633A Please enter the CUU of the
        BDISK disk.

300
ARI0647D Do you want to do a CMS FORMAT/RESERVE command on disk 300?
        Enter 0(No) or 1(Yes).

yes
:
ARI0311I You may need to modify the SQLDBA SQLDBGEN A file.
        This SQL/DS EXEC updates nonrecoverable
        storage pool control statements only.

ARI6191D Do you want to modify the SQLDBA SQLDBGEN file?
        Enter 0(No), 1(Yes), or 111(Quit).

no
:
ARI0919D Database generation invoked.
        The database will be formatted and the original
        database destroyed.

        Enter either:
            DBGEN   to continue, or
            CANCEL  to cancel.

dbgen
:
ARI0717I Start SQLSTART EXEC: 07/19/91 15:21:34 EDT.
ARI0663I FILEDEFS in effect are:
SYSIN   DISK   SQLDBA   SQLDBGEN Q1
SYSPRINT TERMINAL
ARISQLLD DISK   TEMSQLLD LOADLIB Q1
ARISQLLD DISK   ARISQLLD LOADLIB T1
BDISK   DISK   300
LOGDSK1 DISK   301
LOGDSK2 DISK   302
DDSK1   DISK   303
:
ARI2022I SQL/DS is using data spaces for the directory.
ARI2015I The storage pool specification input file is
        empty or not defined. SQL/DS will use the
        default values.
ARI2026I Some or all storage pools will use data spaces.
ARI0529I Catalog generation is completed.
:
ARI0796I End SQLDBGEN EXEC: 07/19/91 15:24:21 EDT

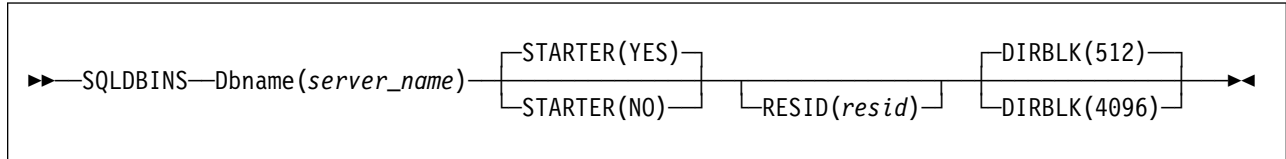
```

SQLDBINS

Generates a new database with a directory disk formatted in either 512 or 4KB blocks by calling SQLDBGEN. It also:

- Creates the DBS Utility package
- Creates the HELP Text tables
- Creates the SYSLANGUAGE table
- Loads the HELP Text tables
- Loads the Sample tables
- Loads the ISQL package
- Loads the FIPS flagger package.

Syntax



DIRBLK

Specifies whether the SQLDBGEN EXEC will generate the new directory in 512 or 4KB blocks. The default is 512. This parameter is only effective if you answer YES to message ARI0647D, which asks whether you want to format the directory disk.

Note: Refer to *DB2 Server for VM System Administration* for a complete description of this EXEC and all its parameters.

Description

This EXEC is updated for VMDSS, and now includes a new parameter **DIRBLK**. You can use this EXEC as described in *DB2 Server for VM System Administration*. Just make sure that if you want to use Data Spaces Support with the directory of your new database, you include the parameter DIRBLK (4096).

Example

To generate a new database with a directory disk at address 300 blocked to 4KB, run the SQLDBINS EXEC with parameter DIRBLK(4096). For example,

```

sqldbins db(SQLDBA) dirblk(4096)
ARI0717I Start SQLDBINS EXEC: 07/19/91 15:02:24 EDT.
ARI06010D Do you want to install English DB2 Server for VM HELP text?
Enter 0(No), 1(Yes), or 111(Quit).

no
ARI0720I Default DB2 Server for VM bootstrap file SQLDBA SQLRMBT created
on the production disk.
ARI0720I Default DB2 Server for VM bootstrap file SQLDBA SQLDBBT created
on the production disk.
ARI0720I Default DB2 Server for VM bootstrap file SQLDBA SQLISBT created
on the production disk.
ARI0721I Get DB2 Server for VM production minidisk READ access: SQLDBA 195.

ARI0717I Start SQLDBGEN EXEC: 07/19/91 15:02:33 EDT.

ARI0633A Please enter the CUU of the
BDISK disk.

300
ARI0647D Do you want to do a CMS FORMAT/RESERVE command on disk 300?
Enter 0(No) or 1(Yes).

yes
:
ARI0717I Start SQLSTART EXEC: 07/19/91 15:04:16 EDT.
ARI0663I FILEDEFS in effect are:
SYSIN DISK SQLDBA SQLDBGEN Q1
SYSPRINT TERMINAL
ARISQLLD DISK TEMSQLLD LOADLIB Q1
ARISQLLD DISK ARISQLLD LOADLIB T1
BDISK DISK 300
LOGDSK1 DISK 301
LOGDSK2 DISK 302
DDSK1 DISK 303
:
ARI0919D Database generation invoked.
The database will be formatted and the original
database destroyed.

Enter either:
DBGEN to continue, or
CANCEL to cancel.

dbgen
:
ARI2022I the database manager is using data spaces for the directory.
:
ARI0666I DB2 Server for VM database installed or serviced successfully.
PRT FILE 0014 SENT FROM SQLDBA PRT WAS 0014 RECS 9018 CPY 001 A HOLD NOKEEP
ARI0660I Line-edit symbols restored:
LINEND=# LINEDEL=␣ CHARDEL=@
ESCAPE=" TABCHAR=ON

ARI0796I End SQLDBINS EXEC: 07/19/91 15:14:19 EDT

```

Appendix B. Storage Pool Specification File Format

This appendix describes the format and syntax of the control file used to tailor VMDSS.

For an overview of storage pool specifications, refer to “Storage Pool Specifications” on page 39. For a step by step description of how to use the storage pool specification file, refer to “Changing Storage Pool Specifications at Startup” on page 39.

File Format

The storage pool specification file must have a fixed record length of 80 characters. It can include three types of lines:

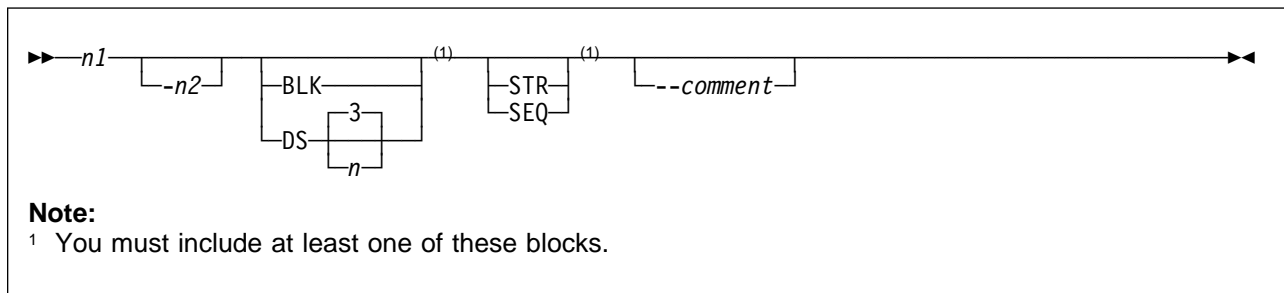
Data Specifies a storage pool or a series of pools, and each pool's VMDSS operating parameters. (See below.)

Blank Allowed anywhere in the file.

Comment Any line that begins with two dashes (--) is a comment line. You can also include a comment at the end of a data line by adding two dashes there. (See below.) The comment ends at the end of the line.

Data Line Syntax

Each data line of the storage pool specification file follows the following syntax:



n1 Specifies that you want to change the specifications for storage pool *n1*. Valid values are integers from 1 to 999.

n2 Specifies a range of storage pools from *n1* to *n2*. Valid values are integers from 1 to 999. *n2* must be greater than or equal to *n1*.

BLK Turns Data Spaces Support off for the storage pools you specify.

DS Turns Data Spaces Support on for the storage pools you specify. This is the default.

- n*** Sets the working storage residency priority of the storage pools you specify to *n*. Valid values are integers from 1 to 5. The default value for *n* is 3.
- STR** Turns on striping for the storage pools you specify. This is the default.
- SEQ** Turns off striping for the storage pools you specify. The database system will allocate pages sequentially on DASD.
- comment*** You can include a comment at the end of the data line. Precede it with two dashes (--).
- Note:** If you do not include a storage pool in the storage pool specification file, the database system will use all the default settings for that storage pool.

Ordering Data Lines

The database manager reads the storage pools specification file from the top down, reading each specification in sequence. It starts with every pool's specifications set to the default values, and updates the current settings with every line it encounters. For example, consider the following specification file:

```
-- Storage Pool Specification File

4-5  SEQ  -- Line 1
1     DS2  -- Line 2
4     DS5  -- Line 3
2-3  BLK  -- Line 4
```

As the database manager reads the file, pools 1 through 5 will all start with Data Spaces Support on, a working storage residency priority of 3, and striping on.

- Line 1** Turns striping off for pools 4 and 5
- Line 2** Sets the working storage residency priority for pool 1 to 2
- Line 3** Sets the priority for pool 4 to 5.
- Line 4** Turns Data Spaces Support off for pools 2 and 3.

You can also achieve the same results with the following specification file:

```
-- Storage Pool Specification File

1     DS2 STR
2     BLK STR
3     BLK STR
4     DS5 SEQ
5     DS3 SEQ
```

While both files are effectively the same, the second file defines each pool without relying on default values and is much easier to read and decipher.

Specification File Example

Consider a database where you want to:

- Turn on striping for storage pools 1 to 10
- Turn off striping for storage pools 11 to 20
- Only use Data Spaces Support for storage pools 2, 5, 10, 11, and 15 to 20
- Set the working storage residency priority to 2 for storage pools 10, 11, and 15 to 19
- Use the default residency priority (3) for pools 2, 5, and 20.

Your storage pool specification file should look like this:

```
-- Storage Pool Specification File
```

```
1    BLK STR
2    DS  STR
3-4  BLK STR
5    DS  STR
6-9  BLK STR
10   DS2 STR
11   DS2 SEQ
12-14 BLK SEQ
15-19 DS2 SEQ
20   DS  SEQ
```

While you could have also coded your file like the following example, you may find it difficult to interpret:

```
-- Storage Pool Specification File
```

```
11-20 SEQ
1    BLK
3-4  BLK
6-9  BLK
12-14 BLK
10-11 DS2
15-19 DS2
```

Appendix C. DB2 Server for VM Initialization Parameters

This appendix describes the additional initialization parameters you can use with VMDSS. As well as a brief description of each command, the table contains page references to other sections of the manual where you can find out more about each one.

Parameter	Understanding	Operating	Measuring and Tuning	Default	Description
MAPPING = L P	15	42	77	L	Sets whether the database manager will use logical or physical mapping. To use logical mapping set MAPPING=L. For physical mapping set MAPPING=P.
SAVEINTV = <i>n</i>	12	42	76	10	Sets the save interval for the database to <i>n</i> blocks of 4KB pages (32 pages in a block). The save interval is the number of blocks in a data space that the database manager modifies before it writes them to DASD. Valid values are integers from 1 to 999999 blocks.
SEPINTDB = Y N	9	42	71, 74	Y	Sets whether the database manager will use separate unmapped internal dbspaces or not. To use unmapped internal dbspaces, set SEPINTDB=Y. Note, setting SEPINTDB=N does not turn Data Spaces Support off. Since mapped internal dbspaces are assigned to one storage pool, they will use whichever DASD I/O system is specified for that pool.
TARGETWS = <i>n</i>	12	42	73	32	Sets the target working storage size for the database to <i>n</i> megabytes. This limits the amount of main and expanded storage that the database machine will use. While the database machine may exceed this target, the database manager will to keep you at or below it if possible. Valid values are integers from 1 to 999999 megabytes.

Example

To explicitly set all the VMDSS initialization parameters to their default values, use the following SQLSTART command:

```
sqlstart db(SQLDBA) parm(targetws=32 saveintv=10 sepintdb=y mapping=L)  
ARI0717I Start SQLSTART EXEC: 05/23/91 09:48:57 EDT.  
ARI0320I The default database name is SQLDBA.  
:  
ARI0015I SEPINTDB parameter value is Y.  
ARI0016I SAVEINTV parameter value is 10.  
ARI0015I MAPPING parameter value is L.  
ARI0016I TARGETWS parameter value is 32.  
:  
:
```

Appendix D. Determining Number of Data Spaces

This appendix describes how to calculate the maximum number of data spaces your database machine will need, and their total size. It also describes how to determine how many spaces your database machine is currently using.

Maximum Number of Data Spaces

To calculate the maximum number of data spaces that your database may use, follow the instructions for the type of mapping you are using (logical or physical). (Logical is the default and suggested type of mapping for most applications.)

Logical Mapping

If you are using logical mapping, you can calculate the maximum number of data spaces by selecting the correct formula from Figure 17. Substitute the total number of public, private, and internal dbspaces in your database into the formula. When appropriate also substitute the number of cylinders in your directory disk and the conversion factor for your directory disk's DASD type. (Use Table 5 on page 92 to look up the directory conversion factor for your database.)

Unmapped Internal Dbspaces?	Data Spaces used with the SQL/DS Directory?	Formula for Maximum Data Spaces
		Note: $\lceil \sim \rceil$ Round up to the nearest integer.
No	No	$\lceil \left(\frac{\sum \text{Public} + \sum \text{Private} + \sum \text{Internal}}{524288} \right) \rceil$
No	Yes	$\lceil \left(\frac{\sum \text{Public} + \sum \text{Private} + \sum \text{Internal}}{524288} \right) \rceil$ + $\lceil \left(\frac{\sum \text{Directory}}{\text{Conversion Factor}} \right) \rceil$
Yes	No	$\lceil \left(\frac{\sum \text{Public} + \sum \text{Private}}{524288} \right) \rceil$ + $\lceil \left(\frac{\sum \text{Internal}}{524288} \right) \rceil$
Yes	Yes	$\lceil \left(\frac{\sum \text{Public} + \sum \text{Private}}{524288} \right) \rceil$ + $\lceil \left(\frac{\sum \text{Internal}}{524288} \right) \rceil$ + $\lceil \left(\frac{\sum \text{Directory}}{\text{Conversion Factor}} \right) \rceil$

Figure 17. Calculating the maximum number of data spaces your database will use with logical mapping.

DASD Type	3350	3375	3380	3390	9345
Conversion Ratio	4369	5461	3495	2912	3495

Example

Consider a database where you are planning to use the directory with data spaces, and unmapped internal dbspaces. It has a total of 640 000 pages allocated to public and private dbspaces, and 80 internal dbspaces of 1024 pages each. It also has a 40-cylinder 3380 directory disk.

Choosing the bottom formula, you would perform the following calculation:

$$\begin{aligned}
& \left\lceil \left(\frac{640000}{524288} \right) \right\rceil + \left\lceil \left(\frac{81920}{524288} \right) \right\rceil + \left\lceil \left(\frac{40}{3496} \right) \right\rceil \\
= & \left\lceil (1.221) \right\rceil + \left\lceil (0.156) \right\rceil + \left\lceil (0.011) \right\rceil \\
= & 2 + 1 + 1 = 4 \text{ Data Spaces}
\end{aligned}$$

Note: If you have a storage constrained environment, please see “Real Storage Requirements for Data Spaces” on page 77.

Physical Mapping

If you are using physical mapping, you can calculate the maximum number of data spaces from the results of a SHOW POOL operator command. Add the total number of pages in each pool to the total number of pages in free areas (deleted dbextents) and substitute this number into the correct formula in Figure 18 on page 94. When appropriate also substitute the number of internal dbspace pages, the number of cylinders in your directory disk, and the conversion factor for your directory disk's DASD type. (Use Table 5 on page 92 to look up the directory conversion factor for your database.)

Unmapped Internal Dbspaces?	Data Spaces used with the SQL/DS Directory?	Formula for Maximum Data Spaces
		Note: $\lceil \sim \rceil$ Round up to the nearest integer.
No	No	$\lceil \left(\frac{\sum \text{Pages from SHOW POOL}}{524288} \right) \rceil$
No	Yes	$\lceil \left(\frac{\sum \text{Pages from SHOW POOL}}{524288} \right) \rceil + \lceil \left(\frac{\sum \text{Directory}}{\text{Conversion Factor}} \right) \rceil$
Yes	No	$\lceil \left(\frac{\sum \text{Pages from SHOW POOL}}{524288} \right) \rceil + \lceil \left(\frac{\sum \text{Internal}}{524288} \right) \rceil$
Yes	Yes	$\lceil \left(\frac{\sum \text{Pages from SHOW POOL}}{524288} \right) \rceil + \lceil \left(\frac{\sum \text{Internal}}{524288} \right) \rceil + \lceil \left(\frac{\sum \text{Directory}}{\text{Conversion Factor}} \right) \rceil$

Figure 18. Calculating the maximum number of data spaces your database will use with physical mapping.

Example

Consider a database where you are not planning to use the directory with data spaces or unmapped internal dbspaces. The following SHOW POOL was performed for the database:

show pool

POOL NO. 1: NUMBER OF EXTENTS = 3

EXTENT NO.	TOTAL PAGES	NO. OF PAGES USED	NO. OF FREE PAGES	NO. OF RESV PAGES	% USED
1	285	274	11		96
2	285	33	252		11
6	741	0	741		0
TOTAL	<u>1311</u>	307	1004	20	23

POOL NO. 2: NUMBER OF EXTENTS = 3

EXTENT NO.	TOTAL PAGES	NO. OF PAGES USED	NO. OF FREE PAGES	NO. OF RESV PAGES	% USED
3	114	2	112		1
4	114	0	114		0
9	114	0	114		0
TOTAL	<u>342</u>	2	340	20	0

POOL NO. 4: NUMBER OF EXTENTS = 1 SHORT ON STORAGE

EXTENT NO.	TOTAL PAGES	NO. OF PAGES USED	NO. OF FREE PAGES	NO. OF RESV PAGES	% USED
10	285	260	25		91
TOTAL	<u>285</u>	260	25	20	91

FREE AREAS: NUMBER OF DELETED EXTENTS = 3

EXTENT NO.	TOTAL PAGES
(2+)	57
5	<u>171</u>
7	<u>228</u>
8	<u>342</u>
END	9552
TOTAL	10350

Maximum number of DBEXTENTS = 64
ARI0065I Operator command processing is complete.

By adding the underlined values you get the total number of data spaces pages required.

```
1311
 342
 285
  57
 171
 228
+ 342
-----
2336 data space pages
```

Since this number is less than 524288, you will only require one data space. (Select the first formula. Divide 2336 by 524288 and round up to the nearest integer.)

Note: If you have a storage constrained environment, please see "Real Storage Requirements for Data Spaces" on page 77.

Maximum Total Size

To determine the maximum total size of the data spaces, multiply the total number of data spaces by 2 gigabytes. For example, if you required 4 data spaces your total size would be:

$$4 \times 2 = 8\text{GB}$$

Remember that this value is the maximum amount of virtual storage that your database machine will use. You do not need to purchase 8GB of main storage.

Displaying Current Data Spaces

You can display information on the current address spaces available for your database machine with the CP QUERY SPACES command. For example, the following command was issued at the operator console:

```
#cp query spaces
CP QUERY SPACES
ASIT          STORAGE  P/S  SPACE IDENTIFICATION
03EF750000000002    17M  PRV  SQLDBA:BASE
03EF7580000000004   29952K PRV  SQLDBA:DIR0000000000
03EF75C0000000003    87M  PRV  SQLDBA:MAP0000000000
03EF7540000000005   200M  PRV  SQLDBA:UNM0000000000
```

The first address space, BASE, is the primary space for the database machine. DIR, MAP, and UNM identify data spaces for the directory, storage pools, and unmapped internal dbspaces respectively. (Remember that a data space is an address space that contains only data. You cannot run programs from a data space; they must first be loaded into a primary address space.)

If the database machine required an additional data space for its storage pools, it would be identified as MAP0000000001. The size of each address space is listed under STORAGE.

Do not be surprised if you do not see all the data spaces you expect. VMDSS creates spaces as it needs them. For example, a data space will only be created for unmapped internal dbspaces when you request a sort that cannot be contained in the local buffer pool.

For more information on this command refer to *VM/ESA: CP Command and Utility Reference*.

Appendix E. Internal Counters

VMDSS contains a set of internal counters that are primarily intended for IBM's use. However, if you have a good understanding of the CP Macros that control data spaces within VM/ESA, you may find the following counters useful.

ESASEG

The number of data space segments mapped. Data spaces are divided into 256-page chunks called segments. A segment is mapped the first time the database manager references a dbspace page that lies in that segment (the first time it references a dbspace, the database manager decides where in the data space the page will be). Mapping a segment consists of defining a mapping between every dbspace page in that segment and its place on disk. Empty pages are not mapped (because there is no disk page allocated for an empty page). Even if all the dbspace pages in a data space segment are empty, the database manager still marks that segment as mapped and increments ESASEG.

MAPSEGP

The number of calls to CP's MAPMDISK DEFINE macro to map the above ESASEG segments. It may take 0 (all pages empty) or many DEFINES per segment, because one DEFINE can only work on a contiguous set of data space pages and database manager must skip empty pages.

REQMAPPG

The number of requests to remap a page. When the database manager gets a new disk page for a dbspace page (either when database manager shadows it or starts using a previously empty dbspace page), it must redefine the data space mapping for that page to point to the new disk location.

MAPPG

The number of MAPMDISK DEFINE macro calls done to satisfy the REQMAPPG requests above. The database manager tries to delay the requests as long as possible in order to group as many as it can into one DEFINE. In the worst case, there will be 1 DEFINE for every request. In the best case, 1 for every 256 requests.

SAVESLD

The number of calls to CP's MAPMDISK SAVE macro to save the contents of a page in the Data Space on DASD.

SAVEMXRN

The sum of "maximum runs." Before issuing a MAPMDISK SAVE macro for one or more blocks of 32 pages, a maximum run is computed. It is the largest number of contiguous blocks of 32 pages that are being saved. SAVEMXRN is an indicator of whether the blocks of pages being saved are consecutively modified (during a dataload for example) and determines the span length to use for the REFPAGE macro following completion of MAPMDISK SAVE macro (refer to SCBLKREF and SCSPNREF).

SAVEBLK

The number of 32 page blocks for which CP's MAPMDISK SAVE macro was issued. The MAPMDISK SAVE macro is issued for modified blocks

of pages at checkpoints and when the number of modified blocks reaches the SAVEINTV threshold.

WAITSLD

The number of times the database manager was forced to wait for a MAPMDISK SAVE macro to save modified pages on DASD. To ensure that all modified data is safely stored on DASD, the checkpoint processing cannot finish until all outstanding MAPMDISK SAVE requests are complete.

REFBLOCK

The number of times a block form of CP's REFPAGE macro was issued telling CP the reference pattern for a group of pages.

REFBPAGE

The number of pages for which a block form of REFPAGE macro was issued.

REFBSPAN

The sum of spans used in block form of REFPAGE macro.

REFLST

The number of times a list form of CP's REFPAGE macro was issued.

REFLPAGE

The number of pages for which a list form of REFPAGE macro was issued.

DIAG10

The number of DIAGNOSE X'10' instructions processed that release the main or expanded storage back to CP.

DIAG10PG

The number of data space pages that have been released using the DIAGNOSE X'10' instruction.

SCDIAG10

The number of blocks of 32 pages for which storage has been released following the completion of MAPMDISK SAVE request.

SCREF

The number of blocks of 32 pages for which a reference pattern was established following the completion of MAPMDISK SAVE request.

SCBLKREF

The number of times a block form of the REFPAGE macro was issued following the completion of MAPMDISK SAVE request.

SCSPNREF

The sum of spans used in the block form of the REFPAGE macro following the completion of the MAPMDISK SAVE request. The size of each span is based on the largest number of consecutive blocks of 32 pages that were (SAVEMXRN counter). As the number of consecutively modified blocks increases (during a dataload for example) the size of the span given to CP's REFPAGE macro also increases.

SAVEGNRL

The number of save requests made by general agents when the SAVEINTV has been reached for a data space.

SAVECHK0

The number of save requests made by the checkpoint agent to flush the page buffers prior to starting checkpoint processing.

SAVECHK1

The number of save requests made by the checkpoint agent during checkpoint processing.

To display the counters, type the following at the operator console:

```
COUNTER INTERNAL *
```

For example:

```
counter internal *
Counter values at DATE='07-23-91' TIME='08:57:51'
ESASEG 429
MAPSEPG 432
REQMAPP 15
MAPP 13
SAVESLD 1
SAVEMXRN 1
SAVEBLK 1
WAITSLD 0
REFBLOCK 860
REFBPAGE 163912
REFBSPAN 68485
REFLST 1580
REFLPAGE 101120
DIAG10 1580
DIAG10PG 101120
SCDIAG10 0
SCREF 0
SCBLKREF 0
SCSPNREF 0
SAVEGNRL 0
SAVECHK0 0
SAVECHK1 0
ARI0065I Operator command processing is complete.
```

To reset the counters to zero, type the following:

```
RESET INTERNAL *
```

For example:

```
reset internal *
Counters reset at DATE='05-23-91' TIME='09:26:15'.
counter internal *
Counter values at DATE='07-23-91' TIME='09:26:51'
ESASEG 0
MAPSEGPG 0
REQMAPPG 0
MAPPG 0
SAVESLD 0
SAVEMXRN 0
SAVEBLK 0
WAITSLD 0
REFBLOCK 0
REFBPAGE 0
REFBSPAN 0
REFLST 0
REFLPAGE 0
DIAG10 0
DIAG10PG 0
SCDIAG10 0
SCREF 0
SCBLKREF 0
SCSPNREF 0
SAVEGNRL 0
SAVECHK0 0
SAVECHK1 0
ARI0065I Operator command processing is complete.
```

Appendix F. Why is the TARGETWS Value Frequently Exceeded?

To understand why the TARGETWS value exceeds the amount of real storage actually used by the database when it is using DB2 Data Space Support (known as VMDSS), let's look at how TARGETWS operates and how VM/ESA manages real storage.

First, remember that CP controls real storage. However, VMDSS can influence how CP manages the real storage that DSS uses (by the TARGETWS value, REFPAGE macro, and RELPAGE macro).

Second, VMDSS does not remember all of the pages that are or have been in real storage. This is partly a trade off—if VMDSS remembered everything, a lot of CPU time would be spent keeping track. In addition, it is not possible for VMDSS to actually know everything that is in real storage. For example, VMDSS does not know what CP does for the database machine. CP may be stealing away pages before VMDSS can release them. This is usually not a problem, and can reduce the database's storage usage.

There is another CP effect that is frequently overlooked. When VMDSS references a data space page that is not in real storage, CP brings in that page from DASD, and may also bring in other pages in a block. In some cases, these are extra pages that the database expects will be needed and has told CP to block together using the REFPAGE macro. However, CP may bring in extra pages based on its own estimation of previous page usage. In both of these cases, VMDSS does not know if CP brought in extra pages or not. Even if the database gives CP some REFPAGE information, CP uses the current system load to decide if it will bring in extra pages or not, and if so, how many.

What this really means is that VMDSS does not know exactly what real storage is being used for which data space pages at any moment in time.

Third, VMDSS has only one method of reducing the usage of real storage—releasing pages using the RELPAGE macro. The RELPAGE macro notifies CP that a specific data space page (or range of pages) is no longer needed and that CP can immediately reuse the real storage frame that currently holds that virtual data space page (assuming CP has not already stolen that page frame). Also, remember that VMDSS uses RELPAGE at specific points when using data space pages.

Most important, is when a data space page is moved from the data space into a database local buffer. Once the page's contents are copied into the local buffer, it can be released. This implies that a page is NOT released unless it is moved into a local buffer. Also (ignoring the extra complications caused by the Working Storage Residency Priorities), page releasing also only occurs when the TARGETWS setting is exceeded.

VMDSS Usage Scenario

Let's take a look at a VMDSS usage scenario to see how both VMDSS and CP react to changing circumstances.

Assume that the system is initially lightly loaded. There is an abundance of real storage available for use, and the database is brought up. Of course, the database storage usage starts small - less than the TARGETWS value. At this point, assume that there is light usage of the database. So, data space pages are referenced, paged in, moved to the local buffers and NOT RELEASED (assuming the Working Storage Residency Priorities are all 3).

As time passes, the storage size of the database increases. In this unconstrained environment, CP may bring in extra pages (beyond those requested by the database). CP will not be stealing pages; there is still free real storage available and no contention for it. Eventually, the database storage size exceeds the TARGETWS value. Now, VMDSS begins to release NEW data space pages as they are used, but it cannot release any of the previously used pages. Note that, at this point, the database storage size still exceeds the TARGETWS, and nothing will be done at this time to reduce this. VMDSS cannot release old pages, CP is not stealing them, and CP may still be bringing in extra pages (there is still free real storage available).

So the storage size continues to increase and continues to exceed the TARGETWS. Eventually the amount of free real storage becomes scarce, and this causes CP to do two things. First, CP will reduce, and finally stop, bringing in extra pages when a page is requested by VMDSS (and will also begin ignoring REFPAGE requests by VMDSS). Second, CP will begin stealing pages away from the database, so that real storage page frames can be reused by other data space pages needed by the database (at this point, the database and the VM system are handling light loads).

However, these actions all have no overall effect on the database storage size. It still greatly exceeds the TARGETWS value, even though VMDSS has been doing RELPAGE's for quite a while now. In general, VMDSS is releasing pages as fast as it requests them, and CP is only stealing away pages as fast as the database needs new ones. The net effect is to simply maintain the database storage size at a constant value, which still exceeds the TARGETWS value.

Now, let's assume that the VM load starts to increase. More real storage will be needed for other users besides the database. CP will begin stealing away more and more of the database's storage for use by other users. Therefore, the database storage size finally begins to decrease. CP is stealing pages and VMDSS is releasing pages faster than VMDSS is requesting them. Eventually, the database storage size will decrease until it is less than the TARGETWS value. Therefore, VMDSS stops releasing pages. At this point, VMDSS is requesting pages and CP is stealing others.

If the VM system load remains relatively constant, the database storage size will remain close to (but usually exceeding slightly) the TARGETWS value because CP will be stealing away pages about as fast as VMDSS can request them.

However, if the VM system load continues to increase, CP will be stealing away pages faster than VMDSS can request them. The database storage size will

continue to decrease and will now remain below the TARGETWS value because VMDSS cannot request pages faster than CP can steal them. Eventually, some minimum database storage size will be reached where the rate of CP stealing pages equals the rate of VMDSS page requests, which will be below the TARGETWS.

In certain storage-constrained environments, CP never seems to be able to prevent VMDSS from taking "too much" storage. In this context, "too much" simply means that other VM users must wait for storage and thus their response time suffers. In these cases, the only solution (without adding extra real storage) is to issue the CP SET SRM MAXWSS n% command to set a system-wide storage size restriction AND to remove the QUICKDSP option from the database user ID. This is a fairly drastic measure because removing QUICKDSP will degrade the database user's response times. If QUICKDSP is not removed, then the "SET SRM MAXWSS" setting will NOT affect the database machine's real storage usage.

Bibliography

This bibliography lists publications that are referenced in this manual or that may be helpful.

DB2 Server for VM Publications

- *DB2 Server for VM Application Programming*, SC09-2661
- *DB2 Server for VM Database Administration*, SC09-2654
- *DB2 Server for VSE & VM Database Services Utility*, SC09-2663
- *DB2 Server for VM Diagnosis Guide and Reference*, LC09-2672
- *DB2 Server for VSE & VM Overview*, GC09-2806
- *DB2 Server for VSE & VM Interactive SQL Guide and Reference*, SC09-2674
- *DB2 Server for VM Master Index and Glossary*, SC09-2666
- *DB2 Server for VM Messages and Codes*, GC09-2664
- *DB2 Server for VSE & VM Operation*, SC09-2668
- *DB2 Server for VSE & VM Quick Reference*, SC09-2670
- *DB2 Server for VM System Administration*, SC09-2657

DB2 Data Spaces Support Publications

- *DB2 Server Data Spaces Support for VM/ESA*, SC09-2675

Related Publications

- *DB2 Server for VSE & VM Data Restore*, SC09-2677
- *DRDA: Every Manager's Guide*, GC26-3195
- *IBM SQL Reference, Version 2, Volume 1*, SC26-8416
- *IBM SQL Reference*, SC26-8415

VM/ESA Publications

- *VM/ESA: General Information*, GC24-5745
- *VM/ESA: VMSES/E Introduction and Reference*, GC24-5837
- *VM/ESA: Installation Guide*, GC24-5836
- *VM/ESA: Service Guide*, GC24-5838
- *VM/ESA: Planning and Administration*, SC24-5750
- *VM/ESA: CMS File Pool Planning, Administration, and Operation*, SC24-5751

- *VM/ESA: REXX/EXEC Migration Tool for VM/ESA*, GC24-5752
- *VM/ESA: Conversion Guide and Notebook*, GC24-5839
- *VM/ESA: Running Guest Operating Systems*, SC24-5755
- *VM/ESA: Connectivity Planning, Administration, and Operation*, SC24-5756
- *VM/ESA: Group Control System*, SC24-5757
- *VM/ESA: System Operation*, SC24-5758
- *VM/ESA: Virtual Machine Operation*, SC24-5759
- *VM/ESA: CP Programming Services*, SC24-5760
- *VM/ESA: CMS Application Development Guide*, SC24-5761
- *VM/ESA: CMS Application Development Reference*, SC24-5762
- *VM/ESA: CMS Application Development Guide for Assembler*, SC24-5763
- *VM/ESA: CMS Application Development Reference for Assembler*, SC24-5764
- *VM/ESA: CMS Application Multitasking*, SC24-5766
- *VM/ESA: CP Command and Utility Reference*, SC24-5773
- *VM/ESA: CMS Primer*, SC24-5458
- *VM/ESA: CMS User's Guide*, SC24-5775
- *VM/ESA: CMS Command Reference*, SC24-5776
- *VM/ESA: CMS Pipelines User's Guide*, SC24-5777
- *VM/ESA: CMS Pipelines Reference*, SC24-5778
- *VM/ESA: XEDIT User's Guide*, SC24-5779
- *VM/ESA: XEDIT Command and Macro Reference*, SC24-5780
- *VM/ESA: Master Index and Glossary*, SC09-2398
- *VM/ESA: Quick Reference*, SX24-5290
- *VM/ESA: Performance*, SC24-5782
- *VM/ESA: Dump Viewing Facility*, GC24-5853
- *VM/ESA: System Messages and Codes*, GC24-5841
- *VM/ESA: Diagnosis Guide*, GC24-5854
- *VM/ESA: CP Diagnosis Reference*, SC24-5855
- *VM/ESA: CP Diagnosis Reference Summary*, SX24-5292
- *VM/ESA: CMS Diagnosis Reference*, SC24-5857

- *VM/ESA: CMS Data Areas and Control Blocks*, SC24-5858
- *VM/ESA: CP Data Areas and Control Blocks*, SC24-5856
- *IBM VM/ESA: CP Exit Customization*, SC24-5672
- *VM/ESA REXX/VM User's Guide*, SC24-5465
- *VM/ESA REXX/VM Reference*, SC24-5770

C for VM/ESA Publications

- *IBM C for VM/ESA Diagnosis Guide*, SC09-2149
- *IBM C for VM/ESA Language Reference*, SC09-2153
- *IBM C for VM/ESA Compiler and Run-Time Migration Guide*, SC09-2147
- *IBM C for VM/ESA Programming Guide*, SC09-2151
- *IBM C for VM/ESA User's Guide*, SC09-2152

Other Distributed Data Publications

- *IBM Distributed Data Management (DDM) Architecture, Architecture Reference, Level 3*, SC21-9526
- *IBM Distributed Data Management (DDM) Architecture, Implementation Programmer's Guide*, SC21-9529
- *VM/Directory Maintenance Licensed Program Operation and User Guide Release 4*, SC23-0437
- *IBM Distributed Relational Database Architecture Reference*, SC26-4651
- *IBM Systems Network Architecture, Format and Protocol*
- *SNA LU 6.2 Reference: Peer Protocols*
- *Reference Manual: Architecture Logic for LU Type 6.2*
- *IBM Systems Network Architecture, Logical Unit 6.2 Reference: Peer Protocols*
- *Distributed Data Management (DDM) List of Terms*
- *IBM Distributed Data Management (DDM) Architecture, Architecture Reference, Level 3*, SC21-9526
- *IBM Distributed Data Management (DDM) Architecture, Architecture Reference, Level 3*, SC21-9526
- *IBM Distributed Data Management (DDM) Architecture, Architecture Reference, Level 3*, SC21-9526

CCSID Publications

- *Character Data Representation Architecture, Executive Overview*, GC09-2207

- *Character Data Representation Architecture Reference and Registry*, SC09-2190

DB2 Server REXSQL Publications

- *DB2 REXX SQL for VM/ESA Installation*, GC09-2660
- *DB2 REXX SQL for VM/ESA Reference*, SC09-2676

C/370 Publications

- *IBM C/370 Installation and Customization Guide*, GC09-1387
- *IBM C/370 Programming Guide*, SC09-1384

Communication Server for OS/2 Publications

- *Up and Running!*, GC31-8189
- *Network Administration and Subsystem Management Guide* SC31-8181
- *Command Reference*, SC31-8183
- *Message Reference*, SC31-8185
- *Problem Determination Guide*, SC31-8186

Distributed Database Connection Services (DDCS) Publications

- *DDCS User's Guide for Common Servers*, S20H-4793
- *DDCS for OS/2 Installation and Configuration Guide* S20H-4795

VTAM Publications

- *VTAM Messages and Codes*, SC31-6493
- *VTAM Network Implementation Guide*, SC31-6494
- *VTAM Operation*, SC31-6495
- *VTAM Programming*, SC31-6496
- *VTAM Programming for LU 6.2*, SC31-6497
- *VTAM Resource Definition Reference*, SC31-6498
- *VTAM Resource Definition Samples*, SC31-6499

CSP/AD and CSP/AE Publications

- *Developing Applications*, SH20-6435
- *CSP/AD and CSP/AE Installation Planning Guide*, GH20-6764
- *Administering CSP/AD and CSP/AE on VM*, SH20-6766
- *Administering CSP/AD and CSP/AE on VSE*, SH20-6767
- *CSP/AD and CSP/AE Planning*, SH20-6770
- *Cross System Product General Information*, GH23-0500

Query Management Facility (QMF) Publications

- *QMF General Information*, GC26-4713
- *QMF VSE/ESA Setup and Usage Guide*, GG24-4196
- *Managing QMF for VSE/ESA*, SC26-3252
- *Installing QMF on VSE/ESA*, SC26-3254
- *QMF Learner's Guide*, SC26-4714
- *QMF Advanced User's Guide*, SC26-4715
- *QMF Reference*, SC26-4716
- *Installing QMF on VM*, SC26-4718
- *QMF Application Development Guide*, SC26-4722
- *QMF Messages and Codes*, SC26-4834
- *Using QMF*, SC26-8078
- *Managing QMF for VM/ESA*, SC26-8219

DL/I DOS/VS Publications

- *DL/I DOS/VS Application Programming*, SH24-5009

COBOL Publications

- *VS COBOL II Migration Guide for VSE*, GC26-3150
- *VS COBOL II Migration Guide for MVS and CMS*, GC26-3151
- *VS COBOL II General Information*, GC26-4042
- *VS COBOL II Language Reference*, GC26-4047
- *VS COBOL II Application Programming Guide*, SC26-4045
- *VS COBOL II Application Programming Debugging*, SC26-4049

- *VS COBOL II Installation and Customization for CMS* SC26-4213
- *VS COBOL II Installation and Customization for VSE* SC26-4696
- *VS COBOL II Application Programming Guide for VSE* SC26-4697

Data Facility Storage Management Subsystem/VM (DFSMS/VM) Publications

- *DFSMS/VM User's Guide*, SC26-4705

Systems Network Architecture (SNA) Publications

- *SNA Transaction Programmer's Reference Manual for LU Type 6.2*, GC30-3084
- *SNA Format and Protocol Reference: Architecture Logic for LU Type 6.2*, SC30-3269
- *SNA LU 6.2 Reference: Peer Protocols*, SC31-6808
- *SNA Synch Point Services Architecture Reference* SC31-8134

Miscellaneous Publications

- *IBM 3990 Storage Control Planning, Installation, and Storage Administration Guide*, GA32-0100
- *Dictionary of Computing*, ZC20-1699
- *APL2 Programming: Using Structured Query Language*, SH21-1056
- *ESA/390 Principles of Operation*, SA22-7201

Related Feature Publications

- *Control Center Installation and Operations Guide for VM*, GC09-2678
- *IBM Replication Guide and Reference*, S95H-0999

Index

Numerics

5648A70 MEMO 21
loading 21

A

ACCESSLIST directory statement 26
ADDRSPACE directory statement 26
agent 50
archive 27
ARISDBMA EXEC 27
arithmetic operator
in syntax diagrams ix
asynchronous
page fault processing 69
writes 71

B

blocking 45, 70
bootstrap module, contents 31
bootstrap package
contents 30, 31
copied to A-disk 31
creating 30
buffer pool 52, 69
build the database manager 28

C

CHANNEL_BUSY measurement 57
check user status 50
checklist
enable 23
pre-enable 23
checkpoint
interval
choosing 75
default value 76
load measurement 60
managing 75
save interval 76
checkpoint interval
understanding 12
choose
in syntax diagrams ix
CMS EXEC 79
conventions
example xi
EXEC xii
syntax diagram notation viii

corrective service 21
counter
CP INDICATE USER
IO 58
READS 57
RES 57
WRITES 58
CP query time
CTIME 58
TOTCPU 58
VTIME 58
directory 14, 49
DIRREAD 59
DIRWRITE 59
PAGEREAD 59
PAGWRITE 59
storage pool performance
Data Spaces Support 64
displaying 49
resetting 50
standard DASD I/O system 61
understanding 14
using 48
unmapped internal dbspace 50
COUNTER operator command 59
COUNTER POOL
Data Spaces Support 64
example 62, 65
operator command 61
standard DASD I/O system 61
CP directory
maximum 91
statement
ACCESSLIST 26
ADDRSPACE 26
MACHINE 26
XCONFIG 26
update 27
CP monitor 57
CP paging
load measurement 57
CP time 58
CPU
effective use 59
creating
bootstrap package 30
CTIME counter 58

D

DASD 57
cache 69

- DASD** (*continued*)
 - channel 57
 - load measurement 59
 - volume 57
- DASD_BY_ACTIVITY** measurement 57
- data space**
 - effective use measurement 66
 - hit ratio 66
- data space DASD read**
 - load measurement 66
- data space paging**
 - load measurement 66
- data spaces**
 - COUNTER POOL 64
 - directory 10
 - internal dbspace 9
 - page fault 51
 - removing 22
 - storage pool 8
 - turning on and off 40
 - understanding 1
- database**
 - archive 27
 - creating new 48
- database machine**
 - DASD requirement 20
 - logoff 26
 - logon 25
 - modes of operation 17
 - overall performance 57
 - overview 17
 - virtual storage 18
- database manager**
 - building 28
 - verifying 28
- DB2 for VM directory**
 - data spaces support 10
 - performance counter 14
 - understanding 10
- DB2 Server for VM directory**
 - reblocking 45
 - using 45
- dbextent**
 - number 77
 - number per device 77
 - size 77
- default**
 - in syntax diagrams x
- directory buffer**
 - effective use measurement 61
- DIRREAD** counter 59
- DIRWRITE** counter 59
- disabling VMDSS** 36
- distribution tape** 27
- DSFAULT** counter 64

- DSPF agent status** 52
- DSREAD** counter 64
- DSWRITE** counter 64
- dynamic storage size management** 70

E

- effective use measurement**
 - CPU 59
 - data space 66
 - description 55
 - directory buffer 61
 - local buffer 60, 63, 66
 - minidisk caching 57
 - overall VMDSS 66
 - standard blocking 63
- elapsed time** 58
- enable** 23
- enable checklist** 23
- enable code** 27
- example**
 - conventions xi
- EXEC**
 - conventions xii
 - SQLCDBEX 79
 - SQLDBGEN 80
 - SQLDBINS 82
 - SQLGENLD 31
 - SQLSTART 33, 34
- existing data**
 - striping 77
- expanded storage** 11

F

- facility, performance reporting** 57
- FB-512 storage devices** 19
- fragment of syntax**
 - in syntax diagrams xi

H

- hit ratio**
 - data space 66
 - description 55
 - directory buffer 61
 - local buffer 60, 63, 66
 - overall VMDSS 66
 - read 57
 - standard blocking 63
- host variable**
 - in syntax diagrams viii

I

- INDICATE USER**
 - command 57

INDICATE USER *(continued)*

example 58

initialization parameter

changing 42
displaying 43
dynamically changing 43
reference 89

installation option

non-XC mode 23
operating system 23
overview 21
production system 22
removing VMDSS 22
saved segment 22
XC mode 23

installation requirement

CP directory 91
DASD 19
database disk 20
database machine 17, 18
database machine DASD 20
dbextent 20
directory 20
FBA 19
hardware 21
log 20
MAINT machine 17, 18
operating system 17
real storage 19
software 18
virtual storage 18
VM paging DASD 19

internal dbospace

mapped
choosing 71
understanding 9
using 44
unmapped
choosing 71
understanding 9
using 44

interval

sampling 56

IO counter 58

IUCV *BLOCKIO request 61

IUCVBIO counter 61

K

keyword

in syntax diagrams viii

L

LBUFLOOK counter 61, 64

link-edit 28

load measurement

checkpoint 60
CP paging 57
DASD 59
DASD channel 57
data space DASD read load 66
data space paging 66
description 55
I/O 59
local buffer looks 66
logical unit of work 60
standard DASD 63
standard local buffer look 63

loading

saved segments 30

local buffer

effective use measurement 60, 63, 66
hit ratio 66

local buffer looks

load measurement 66

log 75

log buffer 75

log disk 75

logical unit of work (LUW)

load measurement 60

LOGMODE startup parameter 75

M

MACHINE directory statement 26

main storage 11

MAINT machine

installation 17
overview 17
virtual storage 18

manual

organization vii
prerequisite viii
who should use vii

mapping

changing 42
choosing 74, 77
initializing 89
understanding 9

measurement tool 56

memo-to-users 21

MINIDISK_CACHE_BY_TIME measurement 57

multiple user mode

starting DB2 Server for VM 33, 34

N

non-XC mode

database machine 17
hardware 21

non-XC mode (*continued*)
SQL/DS directory 72
verify 33

O

operating system

non-XC mode 17
requirement 17

operator command

COUNTER 59
COUNTER POOL 61

optional

default parameter
in syntax diagrams x
installation steps 36
item
in syntax diagrams ix
keyword
in syntax diagrams x

overall VMDSS

effective use measurement 66
hit ratio 66

overview

checkpoint interval 12
data space 1
database machine 17
directory 10
internal dbospace 9
logical mapping 15
MAINT machine 17
operating system 17
performance counter 14
physical mapping 15
save interval 12
storage management 11
storage pool 8
storage pool structure 14
striping 13
target working storage size parameter 12
working storage residence priority 12

P

page fault processing, asynchronous 69

PAGEREAD counter 59

PAGING_BY_TIME measurement 57

PAGWRITE counter 59

parentheses

in syntax diagrams ix

path length 69

percentage

DASD accesses, read 63
data spaces, read 66

performance

improving 1

performance (*continued*)

measurement 55
reporting facility 57

performance tuning

asynchronous processing 69
blocking 70
buffer pool 69
checkpoint 75, 76
checkpoint interval 75
DASD cache 69
data space 69
dynamic storage management 70
internal dbospace 71
mapping 77
more asynchronous writes 71
path length 69
prefetching 70
save interval 76
SQL/DS directory 72
storage pool 71
storage size 72
striping 70, 76
target working storage size 73

PGREAD counter 61

PGWRITE counter 61

planning

storage pool structure 14

post installation 36

pre-enable checklist 23

prefetch

improving performance 70

prerequisite manual viii

preventive service 21

process 50

processor requirement 21

program directory

loading 21

PTF tape 21

punctuation mark

in syntax diagrams ix

Q

query set 33, 34

QUERY TIME

command 57
example 58

R

ratio

data spaces read-to-write 66
read-to-write 63

read hit ratio 57

read percentage 63, 66

read-to-write ratio 63, 66
READS counter 57
 reblocking directory 45
 relative performance measurement 55
repeat symbol
 in syntax diagrams x
reporting facility, performance 57
required item
 in syntax diagrams ix
requirements
 real storage 77
RES counter 57
resave the saved segment 29
reserved words
 SQL xii
RESET operator command
 example 59
RESET POOL
 example 62, 65
resetting counters 50
restarting
 the application server 32

S

sampling interval 56
save interval
 changing 42
 choosing 76
 initializing 89
 understanding 12
save list definition 51
saved segment
 resaving 22
saved segments
 loading 30
saving
 segments 30
see='Appendix F'.target working storage size 101
see='Appendix F'.working storage size 101
segment saving 30
separate internal dbspace
 changing 42
 initializing 89
service
 corrective 21
 preventive 21
SET POOL 41
SET SAVEINTV 43
SET TARGETWS 43
shadow page 75
shorter path length 69
SHOW ACTIVE 51
SHOW BUFFERS 52
SHOW LOCK ACTIVE 51

SHOW LOCK GRAPH 51
SHOW SAVEINTV 43
SHOW TARGETWS 43
software requirement 18
specification file
 storage pool
 changing 39
 format 85, 87
 identifying 39
 writing 86
SQLCDBEX EXEC 79
SQLDBGEN EXEC 80
SQLDBINS EXEC 82
SQLGENLD EXEC 31
SQLMACH machine
 DASD requirement 20
 modes of operation 17
 overview 17
 virtual storage 18
SQLSTART 89
standard blocking
 effective use 63
standard DASD
 load measurement 63
standard local buffer look
 load measurement 63
status of user 50
storage
 main and expanded 11
storage pool
 Data Spaces Support 40
 performance 71
 specification
 checking 41
 dynamically changing 41
 specification file
 changing 39
 example 87
 format 85
 identifying 39
 writing 86
storage pool
 planning structure 14
 understanding 8
storage pool counter
 data space
 DSFAULT 64
 DSREAD 64
 DSWRITE 64
 LBUFLOOK 64
 standard
 IUCVBIO 61
 LBUFLOOK 61
 PGREAD 61
 PGWRITE 61

striping

- choosing 76
- dbextent size 77
- existing data 77
- number of dbextents 77
- one dbextent 77
- performance 70
- physical device 77
- turning on an off 41
- understanding 13

syntax diagram

- notation conventions viii

system process 50

T

target working storage size

- changing 42
- choosing 73
- initializing 89
- understanding 12

tool

- measurement 56

TOTCPU counter 58

U

user agent 50

user status 50

USER_RESOURCE_UTIL 57

V

verify the database manager 28

virtual disk

- storage pools 8

virtual machine time 58

virtual storage

- requirement 18

VM system paging DASD

- internal dbspace 9
- requirement 19

VM/ESA

- requirement 17

VTIME counter 58

W

working storage residence priority

- changing 40
- choosing 73
- understanding 12

working storage size

- improving performance 70
- management 72

workload 55

WRITES counter 58

writing asynchronously 71

X

XC mode

- CP directory statement 26
- database machine 17
- hardware 21
- SQL/DS directory 72
- verify 34

XCONFIG directory statement 26

Communicating Your Comments to IBM

DB2® Server for VM
Data Spaces Support for VM/ESA
Version 6 Release 1
Publication No. SC09-2675-00

If there is something you like—or dislike—about this book, please let us know. You can use one of the methods listed below to send your comments to IBM. If you want a reply, include your name, address, and telephone number. If you are communicating electronically, include the book title, publication number, page number, or topic you are commenting on.

The comments you send should only pertain to the information in this book and its presentation. To request additional publications or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give it to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
 - United States and Canada: 416-448-6161
 - Other countries: (+1)-416-448-6161
- If you prefer to send comments electronically, use the network ID listed below. Be sure to include your entire network address if you wish a reply.
 - Internet: torrcf@ca.ibm.com
 - IBMLink: [toribm\(torrcf\)](mailto:toribm(torrcf)@ca.ibm.com)
 - IBM/PROFS: [torolab4\(torrcf\)](mailto:torolab4(torrcf)@ca.ibm.com)
 - IBMMAIL: [ibmmail\(caibmwt9\)](mailto:ibmmail(caibmwt9)@ca.ibm.com)

Readers' Comments — We'd Like to Hear from You

**DB2® Server for VM
Data Spaces Support for VM/ESA
Version 6 Release 1
Publication No. SC09-2675-00**

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Complete	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to find	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Easy to understand	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Well organized	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Applicable to your tasks	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Canada Ltd. Laboratory
Information Development
2G/345/1150/TOR
1150 EGLINTON AVENUE EAST
NORTH YORK ONTARIO CANADA M3C 1H7

Fold and Tape

Please do not staple

Fold and Tape



File Number: S370/4300-50
Program Number: 5648-A70



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC09-2675-00



Spine information:



DB2® Server for VM

Data Spaces Support for VM/ESA

Version 6 Release 1