

# **FOCUS for S/390**

PowerBook

# Contents

- 1 Reducing Overhead .....1-1**
  - Producing Multiple Outputs in One Pass of a Data Source (Pooled Tables)..... 1-2
    - Overview ..... 1-2
    - Sub Pool Boundaries and Pooling Restrictions ..... 1-7
    - Estimating Memory Requirements ..... 1-9
    - Memory Requirements ..... 1-11
    - Sharing Selection Criteria and Filters Across Requests in a Pool ..... 1-13
    - Criteria When Pooling non-Relational Database Requests ..... 1-13
    - Criteria When Pooling Relational Database Requests ..... 1-13
    - Criteria When Pooling Batch Requests ..... 1-14
    - Selecting a Sort Utility ..... 1-15
    - Observing the Results of Pooling (TRACEON)..... 1-15
    - Installing the Pooled Tables Option ..... 1-18
  - Aggregating and Sorting Columns in One Pass of a Data Source (BY TOTAL)..... 1-20
  
- 2 Expediting Retrievals and Processing.....2-1**
  - Doing Keyed Retrievals from FOCUS HOLD Files ..... 2-2
  - Intelligent Partitioning of Large (2GB) Data Sources ..... 2-5
    - Intelligent Partitioning..... 2-5
  - Selecting Optimal Retrieval Paths Automatically (AUTOPATH) ..... 2-15
  - Concatenating Dissimilar Data Sources ..... 2-17
    - Concatenating Multiple Sources - the MORE Command ..... 2-17
  - Using External Indexes for FOCUS Data Sources ..... 2-25
    - Creating an External Index ..... 2-25
    - Concatenating Index Data Sources..... 2-26
    - Positioning Indexed Fields ..... 2-26
    - Adding a New Data Source to an Existing Index Data Source ..... 2-31
    - Error Messages Associated with External Indexes..... 2-36
  - Using Indexed Fields for Retrieval Automatically (AUTOINDEX) ..... 2-36
  - Using External Sorts to Expedite Large Reports ..... 2-39
    - External Sort Products Supported ..... 2-40
    - Displaying External Sort Messages..... 2-40
  - Using External Sorts to Optimize Aggregation ..... 2-41
  - Using External Sorts to Expedite Production of HOLD Files ..... 2-43
    - Conditions for Use ..... 2-43

<b>3</b>	<b>Exploiting System Enhancements and Raised Limits .....</b>	<b>3-1</b>
	Improving Page Handling With TRACKIO .....	3-2
	Reducing I/O With MINIO Database Access Method .....	3-3
	MINIO Usage .....	3-3
	Employing the IBI MVS Subsystem .....	3-6
	<b>Index .....</b>	<b>I-1</b>

Cactus, EDA, FIDEL, FOCCALC, FOCUS, FOCUS Fusion, Information Builders, the Information Builders logo, SmartMode, SNAPpack, TableTalk, and Web390 are registered trademarks and Parlay, SiteAnalyzer, SmartMart, WebFOCUS, and WorldMART are trademarks of Information Builders, Inc.

Acrobat and Adobe are registered trademarks of Adobe Systems Incorporated.

NOMAD is a registered trademark of Aonix.

UniVerse is a registered trademark of Ardent Software, Inc.

IRMA is a trademark of Attachmate Corporation.

Baan is a registered trademark of Baan Company N.V.

SUPRA and TOTAL are registered trademarks of Cincom Systems, Inc.

Impromptu is a registered trademark of Cognos.

Alpha, DEC, DECnet, NonStop, and VAX are registered trademarks and Tru64, OpenVMS, and VMS are trademarks of Compaq Computer Corporation.

CA-ACF2, CA-Datcom, CA-IDMS, CA-Top Secret, and Ingres are registered trademarks of Computer Associates International, Inc.

MODEL 204 and M204 are registered trademarks of Computer Corporation of America.

Paradox is a registered trademark of Corel Corporation.

StorHouse is a registered trademark of FileTek, Inc.

HP MPE/iX is a registered trademark of Hewlett Packard Corporation.

Intel is a registered trademark of Intel Corporation.

ACF/VTAM, AIX, AS/400, CICS, DB2, DRDA, Distributed Relational Database Architecture, Informix, IBM, MQSeries, MVS, OS/2, OS/400, RACF, RS/6000, S/390, VM/ESA, and VTAM are registered trademarks and DB2/2, HiperSpace, IMS, MVS/ESA, QMF, SQL/DS, VM/XA and WebSphere are trademarks of International Business Machines Corporation.

INTERSOLVE and Q+E are registered trademarks of INTERSOLVE.

Orbit is a registered trademark of Iona Technologies Inc.

Approach and DataLens are registered trademarks of Lotus Development Corporation.

ObjectView is a trademark of Matesys Corporation.

ActiveX, FrontPage, Microsoft, MS-DOS, PowerPoint, Visual Basic, Visual C++, Visual FoxPro, Windows, and Windows NT are registered trademarks of Microsoft Corporation.

Teradata is a registered trademark of NCR International, Inc.

Netscape, Netscape FastTrack Server, and Netscape Navigator are registered trademarks of Netscape Communications Corporation.

NetWare and Novell are registered trademarks of Novell, Inc.

CORBA is a trademark of Object Management Group, Inc.

Oracle is a registered trademark and Rdb is a trademark of Oracle Corporation.

PeopleSoft is a registered trademark of PeopleSoft, Inc.

INFOAccess is a trademark of Pioneer Systems, Inc.

Progress is a registered trademark of Progress Software Corporation.

Red Brick Warehouse is a trademark of Red Brick Systems.

SAP and SAP R/3 are registered trademarks and SAP Business Information Warehouse and SAP BW are trademarks of SAP AG.

Silverstream is a trademark of Silverstream Software.

ADABAS is a registered trademark of Software A.G.

CONNECT:Direct is a trademark of Sterling Commerce.

Java, JavaScript, NetDynamics, Solaris, SunOS, and iPlanet are trademarks of Sun Microsystems, Inc.

PowerBuilder and Sybase are registered trademarks and SQL Server is a trademark of Sybase, Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

Allaire and JRun are trademarks of Allaire Corporation.

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2001, by Information Builders, Inc. All rights reserved. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Printed in the U.S.A.

# Preface

The FOCUS for S/390 PowerBook is a reference for FOCUS customers interested in making their FOCUS applications run faster and more efficiently. Addressed to both FOCUS Systems Administrators and FOCUS users, it describes key performance features added in the 7.0 series of FOCUS releases and provides examples of their use and suggestions for optimal implementation.

We urge sites running versions of FOCUS prior to 7.0, to scan the contents and consider improvements they could implement by simply upgrading to the latest release. While business systems that provide correct output are usually ignored, it's quite possible you could significantly improve your day-to-day operations, freeing up vital resources for other pressing requirements!

## How This Manual Is Organized

This manual includes the following chapters and sections:

Chapter		Contents
1	<i>Reducing Overhead</i>	This chapter presents new ways to get more done in one pass of your data source: <ul style="list-style-type: none"><li>• Producing multiple outputs in one pass of a data source (Pooled Tables)</li><li>• Aggregating and sorting columns in one pass of a data source (BY TOTAL)</li></ul>
2	<i>Expediting Retrievals and Processing</i>	This chapter presents enhancements for expediting retrievals and processing: <ul style="list-style-type: none"><li>• Doing keyed retrievals from FOCUS HOLD files</li><li>• Intelligent partitioning of large (2GB) data sources</li><li>• Selecting optimal retrieval paths automatically (AUTOPATH)</li><li>• Concatenating dissimilar data sources</li><li>• Using external indexes for FOCUS data sources</li><li>• Using indexed fields for retrievals automatically (AUTOINDEX)</li><li>• Using external sorts to expedite large reports</li><li>• Using external sorts to optimize aggregations</li><li>• Using external sorts to expedite production of HOLD files.</li></ul>

Chapter		Contents
3	<i>Exploiting System Enhancements and Raised Limits</i>	<p>This chapter introduces system enhancements for simplifying and expanding operations and lowering operating costs:</p> <ul style="list-style-type: none"> <li>• Improving page handling with TRACKIO</li> <li>• Reducing I/O with MINIO database access method</li> <li>• Employing the IBI MVS Subsystem</li> </ul>

Please note that references to MVS throughout this document apply equally to all supported versions of the OS/390 and MVS operating environments, unless specifically stated otherwise in the text.

## Documentation Conventions

The following conventions are used to describe command syntax in this manual:

<code>THIS TYPEFACE</code>	Denotes a command that you must enter in uppercase, exactly as shown.
<code>this typeface</code>	Denotes a value that you must supply.
<code><u>underscore</u></code>	Indicates a default setting.
Punctuation	Required as shown.
{ }	Indicates two choices from which you must choose one. You type one of these choices, not the braces.
[ ]	Indicates a group of optional parameters. None are required, but you may select one. Type only the information within the brackets, not the brackets.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the information, not the ellipsis points (. . .).
. . . .	Indicates that there are (or could be) intervening or additional commands.

## Related Publications

See the Information Builders Publications Catalog for the most up-to-date listing and prices of technical publications, plus ordering information. To obtain a catalog, contact the Publications Order Department at (800) 969-4636.

You can also visit our World Wide Web site, <http://www.informationbuilders.com>, to view a current listing of our publications and to place an order.

## Customer Support

Do you have questions about FOCUS?

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 9:00 a.m. and 8:00 p.m. EST to address all your FOCUS questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (xxxx.xx) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system, and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of [www.informationbuilders.com](http://www.informationbuilders.com) also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

## Information You Should Have

To help our consultants answer your questions most effectively, please be ready to provide the following information when you call:

- Your six-digit site code number (xxxx.xx).
- The FOCEXEC procedure (preferably with line numbers).
- Master file with picture (provided by CHECK FILE).
- Run sheet (beginning at login, including call to FOCUS), containing the following information:
  - ? RELEASE
  - ? FDT
  - ? LET
  - ? LOAD
  - ? COMBINE
  - ? JOIN
  - ? DEFINE
  - ? STAT
  - ? SET/? SET GRAPH
  - ? USE
  - ? TSO DDNAME OR CMS FILEDEF
- The exact nature of the problem:
  - Are the results or the format incorrect; are the text or calculations missing or misplaced?
  - The error message and code, if applicable.
  - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, FOCUS, your security system, or an interface system changed?
- Is this problem reproducible? If so, how?

- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two databases, have you tried executing a query containing just the code to access the database?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?
- Have any PTFs or patches been installed? This can be determined by issuing the ? [PTF](#) command.

## User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Document Enhancement Request Form on our Web site, <http://www.informationbuilders.com>.

Our thanks, in advance, for your comments.

## Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

---

## CHAPTER 1

# Reducing Overhead

### Topics:

- Producing Multiple Outputs in One Pass of a Data Source (Pooled Tables)
- Aggregating and Sorting Columns in One Pass of a Data Source (BY TOTAL)

This chapter presents two ways to economize on resources and get more done in one pass of your data source. These features can dramatically reduce database I/O, and CPU and elapsed times.

# Producing Multiple Outputs in One Pass of a Data Source (Pooled Tables)

The Pooled Tables option permits you to produce many reports or extract files in a single pass of your data source, dramatically reducing database I/O, CPU and elapsed time. Requests against any data source, file, or JOIN structure that FOCUS reads can be pooled without incurring a penalty - even if the application does not exploit the feature.

Pooling is added with several SET statements and its analytical functions can automatically identify reports that can share database I/O and run them concurrently.

Pooling is applicable whenever consecutive report requests run against the same database, which is ideal for large batch operations, as well as canned FOCUS reporting and data-extract applications. It also applies in most reporting situations where record-selection costs exceed the costs for report formatting.

## Overview

To implement Pooled Tables in an application, you simply add several SET commands - no other changes are required. As FOCUS runs a group of report requests, it starts pooling as soon as a `SET POOL=ON` statement is encountered and pooling continues until it reads a `SET POOL=OFF`. During processing, FOCUS searches for consecutive TABLE requests that access the same file with the same access method and it stores those in sub pools. A read-ahead feature even crosses FOCEXEC boundaries, dividing commands into retrieval and non-retrieval sub pools. These sub pools are collections of TABLE requests and related commands against common data sources - only report requests within sub pools can be combined. Sub-pool boundaries are established whenever FOCUS encounters commands that either alter the data or change the processing environment (see *Sub Pool Boundaries and Pooling Restrictions* on page 1-7).

Sub pools are further subdivided into clusters, which are sets of consecutive TABLE requests against the *same logical database that employ the same access method*. Requests that cannot be pooled due to syntactical or environmental conditions are executed as single-TABLE clusters, which execute concurrently and share their data retrieval and screening processes (as well as related overhead), but do not share sorts or output formatting functions.

## Syntax

### Activating the Pooled Tables Feature

To activate Pooled Tables, issue the following command

```
SET POOL = {OFF|ON}
```

where:

OFF

Ends Pooled Tables and executes any queued requests.

ON

Begins Pooled Tables.

Issue the following command in a report request to supply an estimate for the number of input records for that report:

```
ON TABLE SET ESTRECORDS m
```

where:

*m*

Is the estimate of the number of records being retrieved for a report.

Assign a global value for each report in a pool with the following command

```
SET ESTRECORDS=m
```

The default value is 100,000.

Issue the following command in your report request to supply your estimate of the number of output lines expected for that report:

```
ON TABLE SET ESTLINES n
```

where:

*n*

Is a user estimate of the number of output lines for a report.

You can assign a global value for each report in a pool with the following command

```
SET ESTLINES=n
```

The default value is 0. If no value is given, Pooled Tables assumes no aggregation and that the number of lines is the same as the number of records.

To limit the amount of memory made available for pooling within a cluster (per user), issue the following command

```
SET POOLMEMORY=n
```

where:

*n*

Is the upper limit in kilobytes of memory that FOCUS may use during any cluster for a user. In MVS, this is memory above the 16-megabyte line. In VM, it represents total virtual memory.

The default value is 16,384 K (16 M). The minimum value is 1,024 K.

To reserve memory for other modules, issue the following command

```
SET POOLRESERVE =n
```

where:

*n*

Is an amount of memory (kilobyte) reserved for other modules that Pooled Tables cannot use.

In VM, the default is 1,024K. In MVS, it is 100K.

To limit the number of concurrent external sorts that can run, issue the following command

```
SET MAXEXTSRSTS=n
```

where:

*n*

Is the maximum number (from 1 to 26) of concurrent external sorts permitted. The default is 26.

In VM, only one version of SyncSort can run concurrently. If you use SyncSort in VM, the value of MAXEXTSRSTS is assumed to be 1.

To control automatic application of pooling for batch processing, issue the following command

```
SET POOLBATCH = {OFF|ON}
```

where:

OFF

Disables automatic use of Pooled Tables for batch processing. This is the default. POOLBATCH can be included in the FOCPARM ERRORS, FOCUS PROFILE, a FOCEXEC, or issued in the SYSIN input stream.

SET POOLBATCH=ON has the effect of automatically setting POOL=ON for batch execution. SET POOLBATCH=OFF will not reverse this setting. To disable pooling when POOLBATCH=ON, issue the command SET POOL=OFF.

ON

Enables automatic use of Pooled Tables for batch processing.

To specify a sort utility for use with Pooled Tables, issue the following command

```
SET SORTLIB = sorttype
```

where:

*sorttype*

Can be one of the following

SYNCSORT	Identifies the external sort utility as SYNCSORT.
DFSORT	Identifies the external sort utility as DFSORT.
VMSORT	Identifies the external sort utility as VMSORT.
MVSMGSS	Identifies the external sort utility as SYNCSORT and its messages are displayed (MVS only).
MVMSGDF	Identifies the external sort utility as DFSORT and its messages are displayed (MVS only).

To direct the trace output, issue the following command

```
SET TRACEON=POOLTABL // {PTTRACE|ddname}
```

where:

PTTRACE

Is the default ddname where trace output is directed.

*ddname*

Is an optional ddname where trace output can be directed.

To turn off the trace facility, issue the following command

```
SET TRACEOFF=POOLTABL
```

where:

```
POOLTABL
```

Ends the Pooled Tables Trace facility.

## Example

### Using Pooled Tables

The following example illustrates the ease of implementing Pooled Tables. Here a small amount of memory is provided for Pooled Tables (4,000K); then pooling is turned on and report size estimates are provided for each report. The report requests will be queued until pooling is turned off. At that time, data will be retrieved only once for all report requests in the pool. They will be executed concurrently and the reports printed one after the other.

```
SET POOLMEMORY = 4000
SET POOL=ON
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME HIRE_DATE BY DEPARTMENT
IF HIRE_DATE GE 820101
ON TABLE SET ESTLINES 10000 AND ESTRECORDS 10000
END
```

DEPARTMENT	LAST_NAME	FIRST_NAME	HIRE_DATE
-----	-----	-----	-----
MIS	JONES	DIANE	82/05/01
	BLACKWOOD	ROSEMARIE	82/04/01
	GREENSPAN	MARY	82/04/01
PRODUCTION	SMITH	RICHARD	82/01/04
	BANNING	JOHN	82/08/01
	IRVING	JOAN	82/01/04
	ROMANS	ANTHONY	82/07/01
	MCKNIGHT	ROGER	82/02/02

```
TABLE FILE EMPLOYEE
PRINT CURR_SAL BY CURR_JOBCODE
IF CURR_JOBCODE EQ 'A$*'
ON TABLE SET ESTLINES 5 AND ESTRECORDS 4000
END
```

CURR_JOBCODE	CURR_SAL
-----	-----
A01	\$9,500.00
A07	\$11,000.00
	\$9,000.00
A15	\$26,862.00
A17	\$29,700.00
	\$27,062.00

```
TABLE FILE EMPLOYEE
PRINT LAST_NAME FIRST_NAME BY DEPARTMENT
IF PAY_DATE FROM 820701 TO 820831
ON TABLE SET ESTRECORDS 120000
END
SET POOL=OFF
```

DEPARTMENT	LAST_NAME	FIRST_NAME
-----	-----	-----
MIS	SMITH	MARY
	JONES	DIANE
	MCCOY	JOHN
	BLACKWOOD	ROSEMARIE
	GREENSPAN	MARY
	CROSS	BARBARA
PRODUCTION	STEVENS	ALFRED
	SMITH	RICHARD
	BANNING	JOHN
	IRVING	JOAN
	ROMANS	ANTHONY
	MCKNIGHT	ROGER

## Sub Pool Boundaries and Pooling Restrictions

Sub pools are collections of TABLE or GRAPH requests and related commands. Only reports within a sub pool can be pooled together to share I/O. Sub-pool boundaries are imposed by non-retrieval commands that can change the data or retrieval method for the data source. Therefore, you cannot reliably pool together reports on either side of a sub-pool boundary. When sub-pool boundary commands are encountered, pooling temporarily halts and all queued requests are executed.

Sub-pool boundaries are created when:

- A FOCEXEC completes execution and control is returned to the command line.
- A -RUN or -EXIT command is issued in a FOCEXEC.
- A DEFINE FILE filename ADD command is issued.
- A non-TABLE or GRAPH command is issued that could change the data (MAINTAIN, MODIFY, SQL), change the source of the data (DYNAM, USE), change the retrieval method (JOIN, PASS, FILTER), or change the operating environment (TSO, MVS, CMS), Table 1 lists retrieval commands included in sub pools. Table 1 lists commands that cause sub-pool boundaries.
- Any SET or ON TABLE SET command that can alter retrieval or the Pooled Tables environment. Table 3 lists SET commands that cause sub pool boundaries. SET commands included in ? SET ALL that are not on this list do not cause sub-pool boundaries.

This list is accurate for FOCUS Version 7.0 Release 8 and subsequent releases, but may be subject to change in future releases.

?	?F	?FF	CHECK	DEFINE
GRAPH	HELP	HOLD	OFFLINE	ONLINE
PCHOLD	REPLOT	RETYPE	SAVB	SAVE
TABLE	TABLEF			

**Table 1. Commands included in sub pools**

ACE	ANALYSE	CALC	CMS	COMBINE
COMPILE	CREATE	DECRYPT	DYNAM	ENCRYPT
EX	EXEC	FILETALK	FILTER	FIN
FINISH	FIXPACK	FS	FSCAN	GRAPHTALK
JOIN	LET	LOAD	MAINTAIN	MATCH
MODIFY	MODIFYTALK	MPAINT	MVS	PASS
PLOTTALK	REBUILD	RECALC	REMOTE	RESTRICT
RUN	SCAN	SET	SQL	TABLETALK
TED	TSO	UNLOAD	USE	WINDOW
XFER				

**Table 2. Commands that cause sub- pool boundaries**

ADABAS	AGRRATIO	ALL.	AUTOINDEX
AUTOPATH	AUTOSTRATEGY	AUTOTABLEF	BINS
BLKCALC	BYPANEL 2	BYSCROLL	CACHE
CALC	CALCMEMORY	CALCROWS	CALCWAIT
CARTESIAN	CDN	COLUMNSCROLL2	COMMIT
COMPUTE	CONSULTOPTN	CURRENCY	DATETIME
DEFCENT	ESTLINES 1	ESTRECORDS 1	EXTSORT
FIELDNAME	FILENAME	FIXRETRIEVE	FOCSTACK
FOC144 1	HIPERFOCUS	HTMLMODE	ICUFORM
IMPLIEDLOAD	IMS	LABELPROMPT	LANGUAGE
LE370	LOADLIMIT	LOOKGRAPH	MAXLRECL
MAXPOOLMEM	MINIO	MODE XXXXXX	MPRINT
PASS	POOL	POOLBATCH	POOLFEATURE
POOLMEMORY	POOLRESERVE	PREFIX	PREVIEW
PRINTPLUS 2	QUALCHAR	RECORDLIMIT 1	SAVEMATRIX 2
SHIFT	SM	SQLENGINE	SQLTCARTES
SQLTOPTTF	STYLEMODE	SUSI	SUTABSIZE
TCPIPINT	TEMP DISK	TERMINAL	TEXTFIELD
TRACKIO	TRMSD	TRMSW	TRMTYP
USER	WINPFKEY	XRETRIEVAL	YRTHRESH
3DGRAPH			

*Table 3. SET commands that cause sub- pool boundaries*

**Note:**  
 1 - Sub pool boundary with SET only  
 2 - Sub pool boundary with ON TABLE SET only

## Reference

### Restrictions - Single TABLE Clusters

In certain instances, reports cannot be pooled due to syntactical or environmental conditions. In this case, they are executed as single TABLE clusters. Reports in this category include:

- TABLEF requests.
- MATCH requests.
- Extended Matrix Reports (EMR).

- Reports using SET ALL=ON or PASS.
- Reports against FOCUS databases using an explicit indexed view or an implicit indexed view, using AUTOINDEX.
- Reports against relational databases where aggregation is passed to the DBMS.
- Reports that use MORE, ON field RECAP, COUNT DISTINCT, DST., INCLUDES, EXCLUDES, or COUNT as a verb object.
- Reports that use a redefined database field.
- Reports issued from the FOCUS command line.
- Reports that use a self-referential Filter or DBA value restriction.
- Reports that have more then 256 values in an equality IF or WHERE test.
- Reports executed when \$ORTPARM is allocated.
- Reports that use a user written subroutine except those in Table 4. Generally, subroutines that require initialization and are then reused cannot be pooled. Random-number generator subroutines are a good example.

ARGLEN	ATODBL	AYM	AYMD	BAR
BITSON	BITVAL	BYTVAL	CHGDAT	CHKFMT
CHKPCK	CTRAN	CTRFLD	DADMY	DADYM
DAMDY	DAMYD	DAYDM	DAYMD	DMOD
DOWK	DOWKL	DTDYD	DTDYM	DTMDY
DTMYD	DTYDM	DTYMD	EXP	FEXERR
FINDMEM	FMOD	FTOA	GETPDS	GETTOK
GETUSER	GREGDT	HEXBYT	HHMSS	IMOD
ITONUM	ITOPACK	ITOZ	JULDAT	LCWORD
LJUST	LOCASE	OVLAY	PARAG	PCKOUT
POSIT	RJUST	SOUNDEX	SUBSTR	TODAY
UFMT	UPCASE	YM		

***Table 4. Subroutines and Functions that can be pooled***

## Estimating Memory Requirements

The number of executable reports per cluster depends on how much memory is allocated to Pooled Tables (POOLMEMORY). To optimize pooling capacity, give POOLMEMORY adequate size: MVS region size, or VM virtual memory. Reduce POOLRESERVE after loading interface and other modules.

To improve the pooling potential of requests, remove unnecessary sub-pool boundary commands such as extraneous -RUN statements and consolidate necessary sub-pool-boundary-forcing commands such as DYNAM and SET statements. You can further improve opportunities for pooling within clusters by grouping requests with the same source, entry point and retrieval method together.

For optimal processing, supply accurate estimates for ESTRECORDS, ESTLINES, and POOLMEMORY for each request. Remember that these estimates apply to each report, not to the aggregate size of the set of reports in the cluster. To calculate these sizes, issue ? STAT and review the statistics.

**Example**

**Displaying Report Statistics**

The statistical report produced by ? STAT is useful in tuning applications that employ Pooled Tables.

STATISTICS OF LAST COMMAND					
RECORDS	=	50000	.		
LINES	-	50000	.		
.			.		
.			.		
.			.		
1. READS	=	250000	.		
.			.		
.			.		
.			.		
2. SUBPOOL	=	1			
3. CLUSTER	=	2	ITERATION	=	1 8.
4. #CLUSTER ITEMS	=	25	#TIER ITEMS	=	16 9.
5. SEQ# IN CLUSTER	=	5	SEQ# IN ITER	=	5 10.
6. ESTIMATED RECS	=	50000	ESTIMATED LINES	=	50000 11.
7. REPORT WIDTH	=	148			

**Note:**  
 The ellipses, this annotated sample shows only information concerning Pooled Tables.

For the report just run we see:

- 1. READS                      Sixteen reports were produced in the first iteration. The next 9 reports are executed during subsequent iterations.
- 2. SUB POOL                 This is the fifth report in this iteration.
- 3. CLUSTER                 ESTLINES was set to 50,000. Compare this with LINES at the top of the output. If a discrepancy exists, correct your ESTLINES value for this report.
- 4. # CLUSTER ITEMS        There are 25 reports in the cluster.

5. SEQ# IN CLUSTER This is the fifth report in the cluster.
6. ESTIMATED RECS ESTRECORDS was set to 50,000. Compare this with RECORDS at the top of the output. If a discrepancy, exists correct ESTRECORDS.
7. REPORT WIDTH The report width is 148 bytes.
8. ITERATION This report was produced in the first iteration.
9. # ITER ITEMS
10. SEQ# IN ITER
11. ESTIMATED LINES

## Memory Requirements

Pooled Tables memory requirements per report vary depending on numbers of records selected, output lines produced and report widths, all of which Pooled Tables calculates. Gather ESTLINES and ESTRECORDS input from:

- The statistical message: NUMBER OF RECORDS IN TABLE= LINES=
- The RECORDS and LINES information available on the ? STAT output.
- Previously gathered information from the &RECORDS and &LINES variables.
- When using ACROSS, ESTLINES = number of lines X number of unique ACROSS columns.
- When using IF TOTAL or WHERE TOTAL, ESTLINES is the number of lines before the TOTAL selection is made.

Generally speaking, the memory requirement for a small summary report roughly equals:

`NUMBER OF LINES OF OUTPUT * REPORT WIDTH.`

For large summary reports and detail reports, use:

`NUMBER OF RECORDS SELECTED * REPORT WIDTH`

In the absence of estimates, Pooled Tables uses the following defaults:

- ESTRECORDS=100000
- ESTLINES=0
- POOLMEMORY=16,384K
- POOLRESERVE=100K(MVS)/1024K (VM).

When ESTLINES is 0, Pooled Tables uses the current value of ESTRECORDS for ESTLINES. While adequate for large extract reports, this will provide minimal benefit if inaccurate.

The minimum value for POOLMEMORY is 1,024 K. You can set a maximum threshold for POOLMEMORY when you install Pooled Tables.

- In MVS, POOLMEMORY represents memory above the 16-megabyte line. You can also control the total amount of memory available from the operating system above the 16 megabyte line by coding REGION=*n*M on your JCL job card, where *n* is greater than 16.
- In VM, POOLMEMORY represents total virtual memory.

You can also set POOLMEMORY from the command line, during FOCUS initialization (in the PROFILE FOCEXEC), or within an application.

POOLRESERVE reserves memory for use by other modules during the Pooled Tables request-parsing and decision-making processes. For example, initial access to SQL/DS requires loading of Information Builders interface code and IBM modules (this memory will not be made available to Pooled Tables).

You can change either at installation time. You can set POOLRESERVE from the command line, during FOCUS initialization (in the PROFILE FOCEXEC), or within your application.

Suggested values for POOLRESERVE are:

```
Running an interface (not in saved segment): 1024 K
Running an interface (in saved segment):      256 K
Using SyncSort as the external sort:          512 K
Using any other sort:                          128 K
```

Memory for such activities is not used in the Pooled Tables case until the common read is executed. After loading these modules you can reduce POOLRESERVE, perhaps to zero. If the Information Builders interface and IBM load modules are stored in a saved segment, you can even reduce POOLRESERVE before executing Pooled Tables.

The Pooled Tables Trace facility (see *Observing the Results of Pooling (TRACEON)* on page 1-15) displays actual memory allocations for each report and the statistics used to calculate it.

When POOLMEMORY is insufficient to execute every request in a cluster simultaneously, Pooled Tables executes them in iterations, producing as many reports as it can in memory in the first iteration and staging data for the remaining reports in a FOCPOOLT work file it creates for this use. The remaining reports are produced from FOCPOOLT in subsequent iterations, so the original data source is still only accessed once at the outset. When a cluster can be produced directly from memory, no FOCPOOLT file is created.

## **Example Using a Temporary FOCPOOLT Work File**

If a cluster contained 30 report requests, each requiring 1 megabyte of memory, and 10 megabytes was all the memory allocated for POOLMEMORY, Pooled Tables would retrieve data for all 30 reports but produce only the first 10 reports directly from memory (the first iteration), writing the records for the remaining 20 reports to the FOCPOOLT work file. In the next iteration, Pooled Tables would read data for the next 10 reports from FOCPOOLT and process those. In a third iteration it would process the data for the final 10 reports.

FOCPOOLT retrievals are more efficient than going back to the data source because the data is pre-screened and formatted, and because Pooled Tables collected accurate record counts (ESTRECORDS) when it wrote records for the second and subsequent iterations to FOCPOOLT. With accurate memory requirements calculated, Pooled Tables performance is highly optimized.

## **Sharing Selection Criteria and Filters Across Requests in a Pool**

Selection statements that appear in every report request in a cluster are automatically applied just once during Pooled Tables retrievals. To qualify, such tests must refer to the same field and apply an equality test (EQ or IS), however the actual values selected need not be the same. For example, if the first report tests `WHERE FISCAL_YEAR EQ 1997` and the second tests for `WHERE FISCAL_YEAR EQ 1998`, Pooled Tables applies the test `WHERE FISCAL_YEAR EQ 1997 OR 1998` during data retrieval. Common selection tests greatly reduce the size of answer sets returned.

Pooled Tables can also evaluate common selection criteria not based on equality tests through the FOCUS Filters feature. Filters permit specification of simple or complex selection tests against a common file for all reports. If, for example, all reports in a cluster use `WHERE DELETE_FLAG NE 'Y'`, you can create a filter with that test.

Alternately, you could change the test to read `WHERE DELETE_FLAG EQ 'N'` so that the common selection command is used in the Pooled Tables common read.

## **Criteria When Pooling non-Relational Database Requests**

Reports against non-relational databases, such as VSAM, IMS, IDMS, FOCUS, and sequential files, must meet several simple criteria to be pooled into one cluster. To qualify, all reports must access the same data source, use the same Master File and share the same access method. All reports in a cluster must also share the same entry point (the reporting view must be from the same segment and, in the case of indexed access, from the same field). Reports against sequential files always meet these criteria and always pool. Reports against joined structures are pooled if they share the same access method to the host file.

## Criteria When Pooling Relational Database Requests

Reports against relational databases, such as UDB (DB2) and SQL/DS, can be pooled into the same cluster when they share several common attributes. Like non-relational files, all reports must access the same Master File from the same entry point. Reports requiring SQL aggregation (that is, the generated SQL statements contain the `GROUP BY` phrase) are not pooled, which assures that the set presented to each report in the pool is accurate. Further, requests against a multi-table relational view must all reference the same tables to be pooled into the same cluster.

If a view contains table A and table B, all reports that reference only fields in table A can be pooled; all reports that reference only fields in table B can be pooled, and all reports referencing fields in both table A and B can be pooled. However, none of the reports in these sets could be pooled with reports from the other sets. This limitation insures that the RDBMS retrieval engine will use the same optimization logic for each report in the set.

Less stringent pooling requirements apply with optimization off (SQL `SET OPTIMIZATION OFF`). Since FOCUS manages the retrieval and aggregation operations in this case, pooling conditions are the same with optimization off as with non-relational databases. Restrictions regarding common accessed tables and SQL aggregation do not apply.

Pooling benefits obtained with optimization off, versus those gained by allowing the RDBMS to optimize retrievals, vary from case to case. For example, a request requiring an area sweep that returns a large answer set (even with optimization), would be a good candidate to pool with other requests if optimization were turned off.

When the interface trace facility is used for a relational database, the SQL generated for each request is echoed. The SQL is generated during the Pooled Tables parsing phase but is not submitted to the RDBMS. Instead, Pooled Tables constructs an internal request to retrieve all data for the cluster. The SQL `SELECT` statements generated for the cluster are echoed in the trace and these are the statements passed to the RDBMS.

SQL `SELECT` statements generated by Pooled Tables are optimized by the RDBMS. Therefore, the best optimization occurs when all requests in a cluster contain the same equality screening conditions or Filters. In such cases the screening tests are included in the SQL and passed to the RDBMS for optimization. Without application of common selections or Filters, it is possible that efficiencies gained through RDBMS optimization could be lost in pooling individual requests. Consider these two requests: the first returns a small answer set based on a selection against a key field named `KEY1`. The second returns a small answer set based on a selection against a key field named `KEY2`. The independent screening conditions are not included in the SQL generated by Pooled Tables, resulting in an area sweep and large answer set for the cluster. If the two tests were included as an `OR` condition in a Filter, the screening operation would be passed to the RDBMS and a much smaller answer set returned to Pooled Tables.

## Criteria When Pooling Batch Requests

Pooled Tables automatically pools batch requests wherever possible if the POOLBATCH SET command is issued in a user's PROFILE or in FOCPARM. "Batch" here, means any non-interactive session. In MVS, this means whenever ddname SYSIN is allocated to a data set. In VM, non-interactive jobs occur when ddname SYSIN is defined (FILEDEF) to a file, FOCUS is invoked with the syntax `FOCUS IN fileid`, or the VM session is running disconnected.

## Selecting a Sort Utility

Pooled Tables chooses an in-memory FOCUS sort or an external sort based on report-size estimates. Normally, the FOCUS sort is used for reports under a megabyte and external sorts in other cases. The limiting factor on concurrently executing sorts is the amount of memory available to Pooled Tables. While Pooled Tables can execute up to 26 external sorts, this is controlled by the MAXEXTSRSTS setting and by how much memory is provided below the 16-megabyte line in MVS. In VM, only one external sort can be executed with SyncSort. When practical, the FOCUS sort is substituted for the external sort when external sorts are limited but memory is available.

## Observing the Results of Pooling (TRACEON)

The Pooled Tables trace facility breaks down pools into sub pools and clusters, warns when memory allocation is insufficient, and displays report statistics. The trace facility shows how pools were executed to help developers tune their applications.

### Syntax

### Starting the Trace

Start the trace by issuing the command:

```
SET TRACEON=POOLTABL.
```

where:

Trace output is routed (default) to the ddname PTTRACE allocated to SYSOUT (MVS) or the terminal (VM).

To change this, select a different ddname and issue the command:

```
SET TRACEON=POOLTABL//ddname.
```

To route output to disk, allocate or FILEDEF ddname PTTRACE (or the optional ddname you selected) to a file with LRECL 160 and disposition of MOD. You can also allocate the ddname to the terminal.

### Syntax

### Stopping the Trace

Stop the trace with the command:

```
SET TRACEOFF=POOLTABL.
```

## Trace Output

These messages indicate sub pool boundary encounters:

```
Sub pool boundary--prior output required as input
Sub pool boundary--FOCUS/SET command
Sub pool boundary--DEFINE ADD
Sub pool boundary--new MASTER name
Sub pool boundary--new DEFINE clears pre-pool DEFINE
This command will run now, outside of pooling:
A DEFINE ADD will run now, outside of pooling.
```

These messages indicate cluster boundary encounters:

```
Cluster boundary--new master name
Cluster boundary--single-table cluster
Cluster boundary--new alternate view
Cluster boundary--new pool flag
Cluster boundary--new pool condition
Cluster boundary--mid-stream DEFINE
Cluster boundary--new entry segment
Cluster boundary--too many verb objects
```

These messages indicate reports that cannot be pooled (single-table clusters):

```
Single-table cluster--REDEFINED real field
Single-table cluster--User subroutine not known safe
Single-table cluster--self-referential DBA/filter
Single-table cluster--INCLUDES/EXCLUDES selection
Single-table cluster--too many test literals
Single-table cluster--complex test on index
Single-table cluster--$ORTPARM allocated
Single-table cluster--REDEFINED constant real field
Single-table cluster--RANKED BY
Single-table cluster--COUNT DISTINCT
Single-table cluster--RECAP
Single-table cluster--COUNT is a verb object
Single-table cluster--indexed view via AUTOINDEX
Single-table cluster--EMR
Single-table cluster--ON TABLE SET
Single-table cluster--TEXT field
Single-table cluster--PREVIEW mode
Single-table cluster--ALL = ON/PASS
Single-table cluster--per message above
Single-table cluster--indexed view for FOCUS database
Single-table cluster--non-poolable interface request
Single-table cluster--too many verb objects
```

These trace messages appear during the creation and execution of clusters and iterations:

```
Building cluster x...
Cluster contains n table(s)
Cluster n dedicated to command x
Clusters built; sub pool contains x cluster(s).
***** Stack before 1st cluster: *****
***** Stack before nth cluster: *****
***** Begin union table *****
**** Stack before nth iteration: ****
```

During the parsing phase of Pooled Tables, the following statistics are displayed for each report. These indicate whether a report request could be pooled and under what conditions. All reports that have the same pooling criteria can be pooled with each other.

```
Entry Segment   : x
Relational Flag  : y
Pool Flag        : z
Condition Length: n
Condition        : c
```

After execution of a pooled report, the output from ? STAT is included in the trace. The entries for TRACKIO and MINIO are included in the output but their values are not populated. In addition, the following statistics are included:

```
TRAVERSAL MTHD =          x          ENTRY SEGMENT =          i
FOCUS SORT MEM =          y1         EXTSORT MEMORY =          y2
ALGORITHM USED =          z
```

The following trace messages indicate limitations imposed on Pooled Tables by users in executing reports under less than the most favorable conditions, based on parameters provided for POOLMEMORY, POOLRESERVE, ESTRECORDS, and ESTLINES or available memory. These messages do not inhibit the execution of Pooled Tables but make it less efficient. To correct these situations, replace the values for ESTRECORDS and ESTLINES with accurate values or increase memory allocated for Pooled Tables.

```
# concurrent external sorts reduced from x to y by below-16M shortage
Minimum sort memory forces iterations
Warning--POOLMEMORY desired = x but only y is available
Warning: actual line count (x) exceeds lines estimate (y) in heavy
aggregation case
Warning: records estimate (x) off by more than 10%-actual record count=y
Warning: lines estimate (x) off by more than 10%-actual line count = y
```

## Installing the Pooled Tables Option

This section provides installation instructions for all systems, IMS, OS/390 and MVS, and VM/CMS.

### Procedure

#### Installation Instructions for all Systems

You enable Pooled Tables for your release of FOCUS by including the following command in FOCPARM

```
SET POOLFEATURE = ON
```

To disable Pooled Tables, include the following command in the FOCPARM file

```
SET POOLFEATURE = OFF
```

If FOCPARM does not contain a SET POOLFEATURE command, FOCUS assumes Pooled Tables is disabled.

The maximum memory above 16 megabytes that can be requested with the SET POOLMEMORY command can be restricted by including the SET MAXPOOLMEM = *n* command in FOCPARM.

To make POOL = ON the default for all batch jobs, include the command SET POOLBATCH = ON. This must follow the SET POOLFEATURE = ON command in FOCPARM.

Each of the commands is also included in member FOCPARM of ERRORS.DATA (MVS) or in the file FOCPARM ERRORS (CMS).

### Procedure

#### Installation Instructions for MVS

Include the POOLFEATURE, POOLBATCH, and MAXPOOLMEM commands in member FOCPARM in ERRORS.DATA as outlined above. Refer to your FOCUS documentation, to change the default allocation for the file FOCPOOLT.

If you use DFSort and try to run more than 10 sorts concurrently DFSort will display this message:

```
ICE149A      DFSORT IS NOT LICENSED FOR USE ON THIS SYSTEM. RETURN CODE 12,  
             REASON CODE 4.
```

This will cause FOCUS to ABEND. Issue the command SET MAXEXTSRSTS=10 to temporarily avoid this symptom. IBM has fixed this problem with APAR OW29152. Order IBM PTF UW41671 if you run SMS Release 1.3. Order IBM PTF UW41672 if you run SMS 1.4.

### Procedure

#### Installation Instructions for VM/CMS

Include the POOLFEATURE, POOLBATCH, and MAXPOOLMEM commands in the file FOCPARM ERRORS as outlined above. Change the value of POOLRESERVE in FOCPARM ERRORS if appropriate for your installation.

## Commands for the FOCPARM file

To configure Pooled Tables, include the following commands in the FOCPARM file.

```
SET POOLFEATURE = {OFF|ON}
```

where:

OFF

Disables Pooled Tables for this FOCUS site.

ON

Enables Pooled Tables for this FOCUS site.

```
SET POOLBATCH = {OFF|ON}
```

where:

OFF

Does not enable automatic use of Pooled Tables for batch processing. This is the default.

POOLBATCH can be included in the FOCPARM ERRORS, FOCUS PROFILE, a FOCEXEC, or issued in the SYSIN input stream.

SET POOLBATCH=ON has the effect of automatically setting POOL=ON for batch execution. SET POOLBATCH=OFF will not reverse this setting. To disable pooling when POOLBATCH=ON, issue the command SET POOL=OFF.

ON

Enables automatic use of Pooled Tables for batch processing.

```
SET MAXPOOLMEM = n
```

where:

n

Sets an upper limit in kilobytes for memory above 16 megabytes that users can allocate in the SET POOLMEMORY command. The default is 32,768 K (32 M) and the minimum is 1,024K.

## Aggregating and Sorting Columns in One Pass of a Data Source (BY TOTAL)

The BY TOTAL feature permits simultaneous aggregation and sorting of numeric report columns in one pass of the data when you use an aggregating display command (SUM). A non-aggregating display command (PRINT) simply retrieves the data.

The output record sequence (ascending or descending) will depend on your query.

### Sorting by Report Column

```
[RANKED] BY [HIGHEST|LOWEST [n] ] TOTAL display field
```

where:

*RANKED*

Adds a column to the report output that identifies a rank number for each row.

*n*

Is the number of sort field values you wish to display in the report.

*display field*

Can be a fieldname, prefix-operator.fieldname or calculated value.

### Example

### Sorting by Report Column

A BY TOTAL field is treated as a display field when the matrix is created. After the matrix is created, the output lines are aggregated and resorted based on all of the sort fields. Then the output is produced.

For example:

```
TABLE FILE MOVIES
  SUM LISTPR
  BY CATEGORY
  BY RATING
  BY HIGHEST 5 TOTAL AVE.WHOLESALEPR AS 'AVE,WHOLESALEPR'
  PRINT WHOLESALEPR
  BY CATEGORY
  BY RATING
  WHERE CATEGORY EQ 'CLASSIC' OR 'MYSTERY'
END
```

The output for the example appears below:

CATEGORY	RATING	AVE WHOLESALEPR	LISTPR	WHOLESALEPR	
-----	-----	-----	-----	-----	
CLASSIC	G	40.99	89.95	40.99	
	NR		16.08	314.76	14.99
					20.00
					15.99
					10.95
					10.95
					9.99
					40.99
					10.95
					10.99
			15.00		
MYSTERY	NR	9.00	19.98	9.00	
	PG	9.00	39.96	9.00	
				9.00	
	PG13	9.00	19.98	9.00	
	R	16.19	155.89	15.99	

### Example

### Sorting by an Aggregated Display Field

Here we see BY HIGHEST TOTAL used to organize the output lines.

```
TABLE FILE VIDEOTRK
SUM FIRSTNAME
BY HIGHEST TOTAL TRANSTOT
BY CUSTID NOPRINT BY LASTNAME
END
```

TRANSTOT	LASTNAME	FIRSTNAME
-----	-----	-----
55.99	WILLIAMS	KENNETH
35.16	HARRIS	JESSICA
31.83	NON-MEMBER	
31.00	CHANG	ROBERT
29.98	O'BRIEN	DONALD
	GREEN	KRISTEN
24.99	JOSEPH	JAMES
21.25	GRANT	WESTON
21.24	HANDLER	EVAN
21.00	GOODMAN	JOHN
19.53	RATHER	MICHAEL
18.98	COLE	ALLISON
17.17	STEWART	MAUDE
16.49	DIZON	JANET

```

16.00 CRUZ IVY
14.23 STANDLER MICHAEL
13.98 RIESLER LESLIE
13.33 DAVIS JASON
13.18 LEVINE JOSHUA
10.99 SPIVEY TOM
8.00 WHITE PATRICIA
5.84 MONROE PATRICK
3.24 GARCIA JOANN
2.50 MONROE CATHERINE
      KURTZ DAVID
1.25 PARKER RICHARD
1.00 KRAMER CHERYL
      HEALD MICHAEL

```

**Example**

**Ranking and Sorting by an Aggregated Display Field**

The following request adds a rank column to the report output, which ranks each displayed sort value:

```

TABLE FILE MOVIES
SUM WHOLESALPR CNT.WHOLESALPR
BY CATEGORY
RANKED BY HIGHEST 2 TOTAL AVE.WHOLESALPR AS 'AVE.WHOLESALPR'
BY RATING
WHERE CATEGORY EQ 'CLASSIC' OR 'FOREIGN' OR 'MUSICALS'
END

```

The output is:

CATEGORY	RANK	AVE.WHOLESALPR	RATING	WHOLESALPR	WHOLESALPR COUNT
CLASSIC	1	40.99	G	40.99	1
	2	16.08	NR	160.80	10
FOREIGN	1	31.00	PG	62.00	2
	2	23.66	R	70.99	3
MUSICALS	1	15.00	G	15.00	1
	2	13.99	PG	13.99	1
			R	13.99	1

---

## CHAPTER 2

# Expediting Retrievals and Processing

### Topics:

- Doing Keyed Retrievals from FOCUS HOLD Files
- Intelligent Partitioning of Large (2GB) Data Sources
- Selecting Optimal Retrieval Paths Automatically (AUTOPATH)
- Concatenating Dissimilar Data Sources
- Using External Indexes for FOCUS Data Sources
- Using Indexed Fields for Retrieval Automatically (AUTOINDEX)
- Using External Sorts to Expedite Large Reports
- Using External Sorts to Optimize Aggregation
- Using External Sorts to Expedite Production of HOLD Files

This chapter presents features that expedite retrievals and processing.

## Doing Keyed Retrievals from FOCUS HOLD Files

Keyed retrievals greatly reduce the I/Os incurred in reading from FOCUS HOLD files. Commonly, users read databases and create extract files using ON TABLE HOLD. They then join the FIX files to other databases to narrow their view of the data. Applying logical keys in FIX files permits qualified searches for all records that match IF/WHERE tests for the first n KEY fields. Retrieval ends when the screening test detects values above the stated upper limit.

Improved performance is obtained through the use of a SEGTYPE= parameter in the Master File as a logical key for sequential files. With FIXRETRIEVE=ON, FOCUS stops retrieving as soon as an equality test based on this field holds true. Without this feature, the interface reads all records in the QSAM file and passes them to FOCUS to apply screening conditions when creating the final report (still true with SET FIXRET=OFF).

### *Syntax*

### How to Activate Keyed Retrieval

```
SET {FIXRETRIEVE|FIXRET} = {OFF|ON}
```

where:

ON

Supports keyed retrieval. ON is the default.

OFF

Does not support keyed retrievals.

### *Example*

### Halting Keyed Retrievals With IF/WHERE Tests

This Master File describes a FIX file with sorted ascending MYKEY values from 1 to 100,000:

```
FILE=SORTED,SUFFIX=FIX,$
SEGNAME=ONE,SEGTYPE=S1,$
  FIELD=MYKEY,MK,I8,I8,$
  FIELD=MFIELD,MF,A10,A10,$
TABLE FILE SORTED
PRINT MFIELD
WHERE MYKEY EQ 100
END
```

With FIXRETRIEVAL=ON, retrieval stops when MYKEY reaches 101, vastly reducing the potential number of I/Os, since you read only 101 of a possible 100,000 records.

When the Master file for your extract file is created, FOCUS inserts a SEGTYPE of either Sn or SHn depending on BY fields in the request. For example, PRINT field BY field creates a HOLDMAST with SEGTYPE=S1. Using BY HIGHEST field creates a Master with SEGTYPE=SH1. Selection criteria may include lists of equality values. For example,

```
{IF|WHERE} MYKEY EQ x OR y OR z
```

IF/WHERE tests may also include range tests. For example,

```
{IF|WHERE} MYKEY IS-FROM x TO y
```

The maximum number of BY fields permitted is 32.

Keep in mind, when adding unsorted records to a sorted HOLD file using this feature that duplicate records are not retrieved as in earlier FOCUS Releases. For example, if the sorted file contained the following 3 records:

```
Key Rec#
1 1200
2 2340
3 4875
```

and you added this record at the bottom of the file:

```
1 1620
```

the new (duplicate) record (Key=1), which would have been displayed by FOCUS releases prior to 7.0.5, is not displayed by FOCUS releases 7.0.5 or higher, if FIXRETRIEVAL=ON, because retrieval ends when Rec# 2340 (Key=2) is encountered.

This same situation applies when concatenating individually sorted files - the concatenated results are not sorted together (you get only data from the first file in the concatenation). This is illustrated in the following example:

## Example

## Losing duplicate records when concatenating presorted files

```
-DEFAULT &USER=' '
-SET &USERID=IF &USER NE ' ' THEN &USER ELSE GETUSER('A8');
-SET &DEPT1 =&USERID '3'.DEPT1.DATA ;
DYNAM ALLOC FILE PAYS SHR REUSE -
DATASET &DEPT1
SET MULTIPATH = COMPOUND
SET NODATA= ' '
TABLE FILE EMPLOYEE
PRINT PAY_DATE SALARY
BY EMP_ID
WHERE PAY_DATE EQ 820831
ON TABLE HOLD AS PAYS
END
-SET &DEPT2 =&USERID '3'.DEPT2.DATA ;
DYNAM ALLOC FILE PAY1 SHR REUSE -
DATASET &DEPT2
TABLE FILE EMPLOYEE
PRINT PAY_DATE SALARY
BY EMP_ID
```

```
IF PAY_DATE NE ' '
WHERE PAY_DATE EQ 820730
ON TABLE HOLD AS PAY1
END
-*
DYNAM ALLOC FILE PAYS DA EMP.DEPT1.DATA SHR REUSE
DYNAM ALLOC FILE PAY1 DA EMP.DEPT2.DATA SHR REUSE
DYNAM CONCAT FILE PAYS PAY1
TABLE FILE PAYS
SUM SALARY BY PAY_DATE
WHERE EMP_ID FROM '071382660' TO '451123478'
IF PAY_DATE NE ' '
END
-* RESULTS

PAY_DATE          SALARY
-----          -
82/08/31          $186,222.00
```

# Intelligent Partitioning of Large (2GB) Data Sources

With the advent of two-gigabyte capacities for FOCUS data sources, a new intelligent partitioning feature was also added, enabling logical FOCUS databases to span up to 500 gigabytes.

It is not necessary to partition your data sources to take advantage of expanded database capacities. As of Version 7.1, FOCUS applications will automatically support increased FOCUS database capacities when the new FOC2GIGDB parameter is set ON.

## Syntax

### Enabling Two-Gigabyte Database Support

Issue the following command in the FOCPARM profile or, if the data source will reside on a FOCUS Database Server, in HLIPROF:

```
SET FOC2GIGDB = {ON|OFF}
```

where:

ON

Enables support for FOCUS data sources larger than one-gigabyte. Note that an attempt to use FOCUS data sources larger than one-gigabyte in a release prior to FOCUS Version 7.1 can cause database corruption.

OFF

Disables support for FOCUS data sources larger than one-gigabyte. OFF is the default value.

The new horizontal partition is a slice of the entire file structure, which means that FOCUS database partitions are no longer subject to the multiplicative effect of LOCATION-style vertical partitioning. Note, however, that the sum of all physical files associated with one FOCUS data source (the sum of all partitions and LOCATION files) cannot exceed 250. The new partitioning allows FOCUS data sources to grow over time, with periodic repartitioning based on changing application requirements.

## Intelligent Partitioning

Intelligent partitioning support allows each horizontal partition to contain the complete database structure for specific data values or ranges of values. It not only allows you to separate the data into up to 250 physical two-gigabyte files, it also allows you to create an Access File describing actual data values in each partition using WHERE criteria. When processing report requests, these selection criteria are compared to WHERE criteria in the Access File to determine which partitions to retrieve.

To determine which applications will benefit most from partitioning, look for ones that employ USE commands to concatenate data sources, or for data that lends itself to separation based on particular data values or ranges of values, such as data stored by month or by department.

Intelligent partitioning works like an intelligent USE. It examines the Access File when processing a report request to determine which partitions to read, whereas the USE command reads every file on the list. This decreases I/O and delivers significant performance benefits.

To exploit intelligent partitioning:

- Edit the Master File and add the ACCESSFILE attribute.
- Create the Access File using a text editor.

**Note:** Concatenation of multiple partitions is only supported for reporting. You must load or rebuild each physical partition separately. You can either create a separate Master File for each partition to reference in the load procedure, or you can use the single Master File created for reporting against the partitioned data source, if you:

- Issue an explicit allocation command to link the Master File to each partition in turn.
- Run the load procedure for each partition in turn.

Report requests automatically read all required partitions without user intervention.

## Specifying an Access File in a FOCUS Master File

To use the partitioning feature, you must edit the Master File and add an ACCESSFILE attribute identifying the name of the Access File.

## Syntax

### Specifying an Access File for a Partitioned FOCUS Database

```
FILENAME=fname, SUFFIX=FOC, ACCESS[FILE]=accessfile,  
.  
.  
.
```

where:

*fname*

Is the file name of the partitioned data source.

*accessfile*

Is the name of the Access File. Note that this can be any valid name.

**Example****Master File for the VIDEOTR2 Partitioned Database**

```

FILENAME=VIDEOTR2,  SUFFIX=FOC,
ACCESS=VIDEOACK,  $
SEGNAME=CUST,      SEGTYPE=S1
  FIELDNAME=CUSTID,    ALIAS=CIN,          FORMAT=A4,      $
  FIELDNAME=LASTNAME,  ALIAS=LN,          FORMAT=A15,     $
  FIELDNAME=FIRSTNAME, ALIAS=FN,          FORMAT=A10,     $
  FIELDNAME=EXPDATE,   ALIAS=EXDAT,       FORMAT=YMD,     $
  FIELDNAME=PHONE,    ALIAS=TEL,          FORMAT=A10,     $
  FIELDNAME=STREET,   ALIAS=STR,          FORMAT=A20,     $
  FIELDNAME=CITY,     ALIAS=CITY,         FORMAT=A20,     $
  FIELDNAME=STATE,    ALIAS=PROV,         FORMAT=A4,      $
  FIELDNAME=ZIP,      ALIAS=POSTAL_CODE,  FORMAT=A9,      $
SEGNAME=TRANSDAT,  SEGTYPE=SH1,  PARENT=CUST
  FIELDNAME=TRANSDATE, ALIAS=OUTDATE,   FORMAT=YYMD,    $
SEGNAME=SALES,    SEGTYPE=S2,  PARENT=TRANSDAT
  FIELDNAME=TRANSCODE, ALIAS=TCOD,       FORMAT=I3,      $
  FIELDNAME=QUANTITY,  ALIAS=NO,         FORMAT=I3S,    $
  FIELDNAME=TRANSTOT,  ALIAS=TTOT,       FORMAT=F7.2S,  $
SEGNAME=RENTALS,  SEGTYPE=S2,  PARENT=TRANSDAT
  FIELDNAME=MOVIECODE, ALIAS=MCOD,       FORMAT=A6,  INDEX=I, $
  FIELDNAME=COPY,      ALIAS=COPY,        FORMAT=I2,     $
  FIELDNAME=RETURNDATE, ALIAS=INDATE,     FORMAT=YMD,    $
  FIELDNAME=FEE,       ALIAS=FEE,          FORMAT=F5.2S,  $

```

**The FOCUS Access File**

The Access File provides comprehensive metadata management for FOCUS data sources and shields end users from the complex file storage and configuration details used for efficient and transparent access to partitioned and distributed databases.

The Access File describes how to locate, concatenate, join, and select the appropriate physical data files for retrieval requests against one or more FOCUS databases. Access Files are optional in retrieval requests against non-partitioned databases with no location files and play no part in data maintenance requests.

Every request names a Master File and uses the declarations in it to access the data source. When the Master File includes an ACCESSFILE attribute, FOCUS reads the named Access File and uses that to locate the correct data files. Each Master File can point to a separate Access File, or several Master Files can point to a common Access File. This flexibility makes it possible to create one Access File that manages database access for an entire application. If the Master File does not contain an ACCESSFILE attribute, FOCUS attempts to satisfy the request with the Master File alone.

Use an Access File to exploit the following database features:

- Horizontal and vertical partitioning. A database can consist of several separate files, or horizontal partitions, each of which contains the database records for a specific time period, region, or other element. Segments can also be stored separately from the rest of the data source (LOCATION files or vertical partitions). The Access File describes how to concatenate the separate data files.
- Joins. If joined files are partitioned, the Access File describes how to concatenate the separate data files in the join.

Access Files are *required* for using intelligent partitioning. Intelligent partitioning places specific data values in each physical partition and employs the Access File to describe the values in each partition. With this information, FOCUS optimizes database access by retrieving only those partitions whose values are consistent with the selection criteria in the request.

**Note:**

On OS/390 the Access File must be a member of a data set concatenated in the allocation for ddname ACCESS. On VM/ESA the Access File must have the file type ACCESS. FOCUS cannot be used as the file type. The Access File has the same DCB attributes as the Master File (LRECL=80, RECFM=FB, BLKSIZE= multiple of LRECL).

## FOCUS Access File Attributes

The Access File can include the following attributes:

Attribute	Synonyms	Description
MASTERNAME	MASTER	A Master File entry.
DATANAME	DATA	The name of the physical file.
WHERE		The WHERE criteria.
LOCATION		A segment location.

Each Access File declaration begins with a MASTERNAME attribute identifying the Master File to which it applies. By including multiple MASTERNAME declarations, you can use one Access File for multiple Master Files, possibly for an entire application.

**Syntax****How to Create an Access File**

```

MASTERNAME filename1
  DATANAME dataname1 [WHERE test1 ;]
    [LOCATION locationnamea DATANAME datanamea]
  .
  .
  .
  DATANAME dataname2 [WHERE test2 ;]
    [LOCATION locationnameb DATANAME datanameb]
  .
  .
  .
MASTERNAME filename2
  .
  .
  .

```

where:

**MASTERNAME**

Is the attribute that identifies the Master File name. MASTER is a synonym for MASTERNAME.

*filename1, filename2*

Are names of Master Files. You can describe unrelated Master Files in one Access File.

**DATANAME**

Is the attribute that identifies a physical file. DATA is a synonym for DATANAME.

*dataname1, dataname2*

Are the fully qualified physical file names of physical partition files, in the syntax native to your operating environment.

*test*

Is a valid WHERE test. The following types of expressions are supported. You can also combine any number of these expressions with the AND operator:

```
fieldname relational_operator value1 [OR value2 OR value3 ... ]
```

```
fieldname FROM value1 TO value2 [OR value3 TO value4 ...]
```

```
fieldname1 FROM value1 TO value2 [OR fieldname2 FROM value3 TO value4 ...]
```

where:

*fieldname, fieldname1, fieldname2*

Are field names in the Master File.

*relational\_operator*

Can be one of the following: EQ, NE, GT, GE, LT, LE.

*value1, value2, value3, value4*

Are valid values for their corresponding fields.

**Note:**

If the test conditions do not accurately reflect the contents of the files, you may get incorrect results from requests.

**LOCATION**

Is the attribute that identifies a separately stored segment.

*locationnamea, locationnameb*

Are the values of the LOCATION attributes from the Master File.

Segment locations must map one-to-one to horizontal partitions.

*datanamea, datanameb*

Are the fully qualified physical file names of the LOCATION files, in the syntax native to your operating environment.

**Example**

**Describing Intelligent Partitions in a FOCUS Access File**

The following Access File illustrates how to define intelligent partitions for the VIDEOTR2 database, in which data is grouped by date.

For MVS:

```
MASTERNAME VIDEOTR2
  DATANAME USER1.VIDPART01.FOCUS
    WHERE TRANSDATE GT 20001231;
  DATANAME USER1.VIDPART00.FOCUS
    WHERE TRANSDATE FROM 19990101 TO 20001231;
  DATANAME USER1.VIDPART98.FOCUS
    WHERE TRANSDATE FROM 19960101 TO 19971231;
```

For CMS:

```
MASTERNAME VIDEOTR2
  DATANAME 'VIDPART01 FOCUS A'
    WHERE TRANSDATE EQ 20001231;
  DATANAME 'VIDPART00 FOCUS A'
    WHERE TRANSDATE FROM 19990101 TO 20001231;
  DATANAME 'VIDPART98 FOCUS A'
    WHERE TRANSDATE FROM 19960101 TO 19971231;
```

## Example

### Describing Intelligent Partitions With LOCATION Files

In the following SALES Master File, the CUSTDATA segment is stored in a separate LOCATION file named MORECUST:

```
FILENAME=SALES, ACCESSFILE=XYZ,$
  SEGNAME=SALEDATA
.
.
.
  SEGNAME=CUSTDATA, LOCATION=MORECUST,$
```

The corresponding Access File (XYZ) describes one partition for 1994 data, and another partition for the 1993 data. Each has its corresponding MORECUST LOCATION file:

For MVS:

```
MASTERNAME SALES
  DATANAME USER1.SALES94.FOCUS
    WHERE SDATE FROM '19940101' TO '19941231';
  LOCATION MORECUST
    DATANAME USER1.MORE1994.FOCUS

  DATANAME USER1.SALES93.FOCUS
    WHERE SDATE FROM '19930101' TO '19931231';
  LOCATION MORECUST
    DATANAME USER1.MORE1993.FOCUS
```

For CMS:

```
MASTERNAME SALES
  DATANAME 'SALES94 FOCUS A'
    WHERE SDATE FROM '19940101' TO '19941231';
  LOCATION MORECUST
    DATANAME 'MORE1994 FOCUS A'
  DATANAME 'SALES93 FOCUS A'
    WHERE SDATE FROM '19930101' TO '19931231';
  LOCATION MORECUST
    DATANAME 'MORE1993 FOCUS A'
```

## Example Using a Partitioned Database

The following illustrates how to use a partitioned database.

```
TABLE FILE VIDEOTR2
PRINT  LASTNAME FIRSTNAME DATE
WHERE  TRANSDATE FROM '19960101 TO 19971231'
END
```

The output is:

LASTNAME	FIRSTNAME	DATE
HANDLER	EVAN	1996
JOSEPH	JAMES	1997
HARRIS	JESSICA	1997
HARRIS	JESSICA	1996
MCMAHON	JOHN	1996
WU	MARTHA	1997
CHANG	ROBERT	1996

While nothing in the request or output signifies that a partitioned database was used, only the second partition is retrieved, reducing I/O and enhancing performance.

## Describing Joined Files

Master Files can describe cross-references to other Master Files. In simple cases, a Master File alone may be sufficient for describing the cross-reference.

If one of the joined files is horizontally partitioned, only that file needs an Access File to implement the join. When both joined files are horizontally partitioned, however, they can both be described in one Access File or they can each be described in a separate Access File to implement the join. Only the host file can have WHERE criteria in the Access File. When the host and cross-referenced file both contain WHERE criteria, the join operation may produce unexpected results.

## Example

### Joining Partitioned Data Sources

Recall that the cross-referenced field in a join must be indexed. If the host file is partitioned, the cross-referenced file must either contain the same number of partitions as the host file or only one partition.

For MVS:

```
MASTERNAME SALES
  DATANAME USER1.NESALES.FOCUS
  DATANAME USER1.MIDSALES.FOCUS
  DATANAME USER1.SOSALES.FOCUS
  DATANAME USER1.WESALES.FOCUS
```

```
MASTERNAME CUSTOMER
  DATANAME USER1.NECUST.FOCUS
  DATANAME USER1.MIDCUST.FOCUS
  DATANAME USER1.SOCUST.FOCUS
  DATANAME USER1.WECUST.FOCUS
```

For CMS:

```
MASTERNAME SALES
  DATANAME 'NESALES FOCUS A'
  DATANAME 'MIDSALES FOCUS A'
  DATANAME 'SOSALES FOCUS A'
  DATANAME 'WESALES FOCUS A'
```

```
MASTERNAME CUSTOMER
  DATANAME 'NECUST FOCUS A'
  DATANAME 'MIDCUST FOCUS A'
  DATANAME 'SOCUST FOCUS A'
  DATANAME 'WECUST FOCUS A'
```

## Reference

### Usage Notes for Partitioning Large FOCUS Data Sources

- Concatenation of multiple partitions in one request is only valid for reporting. To MODIFY or REBUILD a partitioned database, you must explicitly allocate and MODIFY, Maintain, or REBUILD one partition at a time.
- To sort FOCUS data sources larger than a gigabyte:
  - On MVS you must explicitly allocate ddname FOCSORT to a temporary file with enough space to contain the data.
  - On VM, you must have enough TEMP space available.
- To REBUILD a FOCUS data source larger than a gigabyte:
  - On MVS you must explicitly allocate ddname REBUILD to a temporary file with enough space to contain the data

On VM you must have enough TEMP space available. It is strongly recommended that you REBUILD/REORG to a new file, in sections, to avoid the need to allocate large amounts of space to REBUILD.

In the DUMP phase, use selection criteria to dump a section of the database. In the LOAD phase, make sure to *add* each new section after the first. To add to a database in MVS you must issue the LOAD command with the following syntax:

LOAD NOCREATE

- If you create a FOCUS data source larger than a gigabyte using HOLD FORMAT FOCUS, on MVS you must explicitly allocate ddnames FOC\$HOLD and FOCSORT to temporary files large enough to hold the data; on VM you must have enough TEMP space available.
- The order of precedence for allocating data sources is as follows:
  1. A USE command in effect has the highest precedence. It overrides an Access File or an explicit allocation for a data source.
  2. An Access File overrides an explicit allocation for a data source.
- A DATASET attribute cannot be used in the same Master File as an ACCESSFILE attribute.

# Selecting Optimal Retrieval Paths Automatically (AUTOPATH)

AUTOPATH dynamically selects optimal retrieval paths for accessing FOCUS files by analyzing the file structures and fields referenced, and choosing the lowest possible segment as the entry point. The feature delivers optimized requests without requiring that users analyze the structure of their FOCUS files or any DEFINE commands currently in effect (either implicit or explicit). To do so, AUTOPATH automates TABLE FILE *ddname.fldname* syntax, where *fldname* is not indexed and initiates physical retrieval at the *fldname* segment.

## Syntax

### Initiating AUTOPATH

AUTOPATH is on by default. To turn it off or back on again, issue the following SET command

```
SET AUTOPATH = {ON|OFF}
```

where:

ON

Turns AUTOPATH on (the default).

OFF

Turns AUTOPATH off.

## Reference

### Special AUTOPATH Conditions

- AUTOPATH is invisible and automatic, although certain FOCUS features (such as LAST and self-referencing virtual fields) restrict its use.
- AUTOPATH displays records in the physical order in which they are stored in the file. For sorted output, specify a BY field in your request.

#### Using AUTOPATH

```
SET AUTOPATH=OFF
TABLE FILE CAR
SUM RCOST BY BODYTYPE
END
```

BODYTYPE	RETAIL_COST
-----	-----
CONVERTIBLE	8,878
COUPE	38,320
HARDTOP	5,100
ROADSTER	6,820
SEDAN	114,086

#### Note:

1. Reads physically through the COUNTRY instances.

2. For each COUNTRY, follows pointer to CAR, from CAR to MODEL, from MODEL to BODYTYPE to sum RCOST BY BODYTYPE.

```
SET AUTOPATH=ON
TABLE FILE CAR
SUM RCOST BY BODYTYPE
END
```

BODYTYPE	RETAIL_COST
-----	-----
CONVERTIBLE	8,878
COUPE	38,320
HARDTOP	5,100
ROADSTER	6,820
SEDAN	114,086

**Note:**

1. Reads physically through the BODYTYPE instances, to sum the RCOST by BODYTYPE, saving all access to the higher level segments in the hierarchy. Equivalent to TABLE FILE CAR.BODYTYPE.

## Concatenating Dissimilar Data Sources

The Universal Concatenation feature permits concatenation of data from multiple dissimilar data sources in a single request. Concatenation of different sources (such as FOCUS, DB2, IMS, VSAM) is possible as long as they share corresponding fields with like formats. Data can also be screened through IF and WHERE tests applied through the MORE command used to effect the concatenation.

Retrievals with MORE are accomplished through a main request and sub requests. The main request defines the data fields, the sort criteria, and the output formats of all data. Sub requests define the files and the data fields being concatenated to the data specified in the main request. During retrieval, FOCUS first gathers data from the main source and then gathers data from sources specified in the sub requests. All data is then sorted together and the output produced as specified in the main request.

### Concatenating Multiple Sources - the MORE Command

The MORE command enables retrieval of data from multiple sources with dissimilar Master Files using TABLE, GRAPH, or MATCH. For example, the EMPLOYEE file and the EXPERSON file below both have Employee information, but share common fields. With MORE, you can concatenate their common data into a single file.

Employee			Report Output	
Emp_id	Curr_sal	DEFINE FILE EXPERSON	<u>Emp_id</u>	<u>Curr_sal</u>
123456789	50.00	EMP_ID/A9 =SSN;	123456789	50.00
		CURR_SAL/D12.2 = WAGE;	987654321	100.00
		END		
		TABLE FILE EMPLOYEE		
		PRINT CURR_SAL		
		BY EMP_ID		
		MORE		
		FILE EXPERSON		
		END		
Experson				
SSN	Wage			
987654321	100.00			

## Syntax

### Using the MORE Command

The syntax is

```
TABLE (GRAPH, MATCH) FILE file1
    main request
MORE
FILE file2
    sub request
MORE
FILE file3
    sub request
MORE
.
.
.
END (RUN)
```

where:

TABLE FILE, GRAPH FILE, MATCH FILE

MORE sub requests must be issued from within TABLE, GRAPH, or MATCH commands.

*file1*

Is the name of the first file.

*main request*

Is a request without an END or RUN statement. Sorting, formatting, aggregation, and calculated values, for all data must be described within this request. IF and WHERE criteria used within the main request screen data only from this source.

MORE

Begins a sub request. There is no limit on the number of sub requests (other than available memory).

FILE *file2*

The next file for concatenation follows the FILE command.

*sub request*

Sub requests may only include the WHERE and IF criteria.

RUN, END

Ends a request.

### Matching Field Names and Formats

All fields referenced in the main request must be virtual fields for files named in the sub requests. Referenced fields include those named in COMPUTE, HEADING, aggregation, and sort operators as well as objects of the PRINT, LIST, SUM, COUNT, WRITE, or ADD commands.

Fields named in the main request must correspond with those in the sub requests, both in name and format. Use DEFINE FILE commands when you need to remap or change sub request field names or formats to match corresponding fields in the main request.

A successful format match means:

<b>Format Type</b>	<b>Correspondence</b>
A	Format type and length must be equal.
I,F,D	Format type must be the same.
P	Format type and scale must be equal.
DATE (new)	Always correspond.
DATE (old)	Edit options must be the same.

**Note:**

TEXT fields cannot be concatenated.

## Example

### Using Concatenated Data in a MORE Request

The three following examples show how to concatenate data with a MORE request. The first TABLE request shows data from EMPDATA. The second TABLE request shows data from the PAYHIST file. The third TABLE request shows how to concatenate data in a single request using MORE.

```
TABLE request 1
DEFINE FILE EMPDATA
NEWID/A11 = EDIT(ID, '999-99-9999');
END

TABLE FILE EMPDATA
HEADING
"CURRENT EMPLOYEE SALARIES"
" "
PRINT CSAL BY NEWID AS 'EMPLOYEE ID'
WHERE CSAL GT 65000
END
-RUN

TABLE request 2
DEFINE FILE PAYHIST
NEWID/A11 = EDIT(SSN, '999-99-9999');
CSAL/D12.2M = NEW_SAL;
END
```

```
TABLE FILE PAYHIST
HEADING
"HISTORICAL EMPLOYEE SALARIES"
" "
PRINT CSAL BY NEWID
WHERE NEW_SAL GT 500
END
```

The output is

```
CURRENT EMPLOYEE SALARIES

EMPLOYEE ID          SALARY
-----
000-00-0030          $70,000.00
000-00-0070          $83,000.00
000-00-0200         $115,000.00
000-00-0230          $80,500.00
000-00-0300          $79,000.00
```

HISTORICAL EMPLOYEE SALARIES

```
NEWID                CSAL
-----
100-10-1689          $842.90
                    $982.90
100-11-9950          $508.75
100-14-2166          $876.45
100-15-5843          $508.75
100-16-2791          $567.89
100-16-4984          $1,236.78
100-17-5025          $734.56
100-18-9299          $567.89
```

```
TABLE request 3
DEFINE FILE EMPDATA
NEWID/All = EDIT (ID, '999-99-9999');
END

DEFINE FILE PAYHIST
NEWID/All = EDIT (SSN, '999-99-9999');
CSAL/12.2M = NEW_SAL;
END

TABLE FILE EMPDATA
HEADING
"EMPLOYEE SALARIES"
" "
PRINT CSAL BY NEWID AS 'EMPLOYEE ID'
WHERE CSAL GT 65000
MORE
FILE PAYHIST
WHERE NEW_SAL GT 500
END
```

The output is

## EMPLOYEE SALARIES

EMPLOYEE ID	SALARY
000-00-0030	\$70,000.00
000-00-0070	\$83,000.00
000-00-0200	\$115,000.00
000-00-0230	\$80,500.00
000-00-0300	\$79,000.00
100-10-1689	\$1,685.80
	\$982.90
100-11-9950	\$508.75
100-14-2166	\$876.45
100-15-5843	\$508.75
100-16-2791	\$567.89
100-16-4984	\$1,236.78
100-17-5025	\$734.56
100-18-9299	\$567.89

## MATCH FILE Considerations

Using MATCH with the MORE command enables you to concatenate unlimited files together and merge the concatenated file with up to six additional sets of concatenated files.

All requirements of the MATCH command must be met in the main request. Refer to the *Creating Reports* manual for a full discussion of the MATCH command.

## Syntax

### Using MATCH File With the MORE Command

The syntax is:

```

MATCH FILE file1
      main request
MORE
FILE file2
      subrequest
MORE
FILE file3
      subrequest
RUN
FILE file4
      main request
[AFTER MATCH merge_phrase]
MORE
FILE file5
      sub request
MORE
FILE file6
      subrequest
RUN
FILE file7
      main request
[AFTER MATCH merge_phrase]
MORE
FILE file8
      subrequest
MORE
FILE file9
      subrequest
END

```

All data concatenated in the first answer set is merged with the data concatenated in the second answer set using the AFTER MATCH merge\_phrase (for example, OLD-OR-NEW) in the second answer set. All merged data from the first and second answer sets, now a HOLD file, is then merged with data concatenated in the third answer set using the AFTER MATCH merge\_phrase in the third answer set. This final set of merged data is placed in a HOLD file.

**Example Using MATCH FILE With MORE****Example 1**

```

DEFINE FILE EMPDATA
CURR_SAL/D12.2M = CSAL;
FIRST_NAME/A10 = FN;
EID/A9 = PIN;
END
_*

MATCH FILE EMPLOYEE
SUM CURR_SAL AS 'CURRENT'
    FIRST_NAME AS 'FIRST'
BY EID AS 'SSN'
_*

MORE
FILE EMPDATA
RUN
_*

FILE TRAINING
PRINT EXPENSES
BY PIN AS 'SSN'
AFTER MATCH HOLD OLD-OR-NEW
END
_*

TABLE FILE HOLD
PRINT *
END

```

**The output is**

SSN	CURRENT	FIRST	EXPENSES
---	-----	-----	-----
000000010	\$55,500.00	DANIEL	2,300.00
000000020	\$62,500.00	MICHAEL	.
000000030	\$70,000.00	LOIS	2,600.00
000000030	\$70,000.00	LOIS	2,300.00
000000040	\$62,500.00	RUTH	3,400.00
000000050	\$54,100.00	PETER	3,300.00
000000060	\$55,500.00	DORINA	.
000000070	\$83,000.00	EVELYN	.
000000080	\$43,400.00	PAMELA	3,200.00
000000080	\$43,400.00	PAMELA	3,350.00
000000090	\$33,000.00	MARIANNE	.
000000100	\$32,400.00	TIM	3,100.00
000000110	\$19,300.00	ANTHONY	1,800.00
000000110	\$19,300.00	ANTHONY	2,500.00
000000110	\$19,300.00	ANTHONY	2,400.00
000000120	\$49,500.00	KATE	2,200.00
000000130	\$62,500.00	MARCUS	.

Example 2

```

DEFINE FILE EMPDATA
CURR_SAL/D12.2M = CSAL;
FIRST_NAME/A10 = FN;
EID/A9 = PIN;
END
-*

MATCH FILE EMPLOYEE
SUM CURR_SAL AS 'CURRENT'
FIRST_NAME AS 'FIRST'
BY EID AS 'SSN'
-*

MORE
FILE EMPDATA
RUN
-*

FILE TRAINING
PRINT EXPENSES
BY PIN AS 'EID'
AFTER MATCH HOLD OLD-OR-NEW
END
-*

TABLE FILE HOLD
PRINT *
END

```

Note the use of an AS phrase to change the answer set.

SSN	CURRENT	FIRST	EID	EXPENSES
---	-----	-----	---	-----
000000010	\$55,500.00	DANIEL	000000010	2,300.00
000000020	\$62,500.00	MICHAEL	000000030	2,600.00
000000030	\$70,000.00	LOIS	000000030	2,300.00
000000040	\$62,500.00	RUTH	000000040	3,400.00
000000050	\$54,100.00	PETER	000000050	3,300.00
000000060	\$55,500.00	DORINA	000000080	3,200.00
000000070	\$83,000.00	EVELYN	000000080	3,350.00
000000080	\$43,400.00	PAMELA	000000100	3,100.00
000000090	\$33,000.00	MARIANNE	000000110	1,800.00
000000100	\$32,400.00	TIM	000000110	2,500.00
000000110	\$19,300.00	ANTHONY	000000110	2,400.00
000000120	\$49,500.00	KATE	000000120	2,200.00
000000130	\$62,500.00	MARCUS	000000140	3,600.00
000000140	\$62,500.00	VERONICA	000000150	3,400.00
000000150	\$40,900.00	KARL	000000160	1,000.00
000000160	\$62,500.00	ROSE	000000180	1,250.00
000000170	\$30,800.00	WILLIAM	000000190	3,150.00

## Using External Indexes for FOCUS Data Sources

Users with only READ access to a local FOCUS data sources can take advantage of a new External Index feature to create index data sources to facilitate indexed retrievals when joining or locating records. External indexes deliver equivalent performance to permanent Indexes for retrieval and analysis operations. And, unlike permanent indexes, they also allow indexing on concatenated FOCUS data sources, indexing on real and virtual fields, and indexing selected records from WHERE/IF tests. External indexes are created as temporary data sets unless pre-allocated to a permanent data set and are not updated as the indexed data changes.

With time stamping of FOCUS data sources in Release 7.0, it is always essential to ensure that time stamps on index data sources are not out of date before using them. Index data sources become outdated when any data source comprising the index is changed through a MODIFY, MAINTAIN, SCAN or FSCAN operation. When records comprising the index change, you must recreate the index data source using the EXTERNAL INDEX option of the REBUILD utility. MODIFY and MAINTAIN commands may not be used to update, include or delete the contents of this data source. If the changes do not affect records used to create the external index data source, you may use the new TIMESTAMP option of the REBUILD utility to make the time stamps match. If the time stamp on the index data source is out of date, the following error message is generated:

```
(FOC976) EXTERNAL INDEX EXPIRED
```

## Creating an External Index

External indexes are created with the enhanced REBUILD command. Internally, REBUILD begins a process that reads the data sources comprising the index, gathers the index information, and creates an index data source containing all field, format, segment, and location information. If your index data source already exists, the add feature can be used to append additional index records to the index data source without completely recreating it.

The REBUILD process solicits information about:

- Whether or not you want to add new records from a concatenated data source to the index data source.
- The name of the external index data source that you want to build.
- The name of the data source from which the index information is obtained.
- The name of the field from which the index is to be created.
- Whether or not you want to position the index field within a particular segment.
- Any valid WHERE or IF record selection tests.

## Concatenating Index Data Sources

The external index feature enables indexed retrievals from concatenated FOCUS data sources, whereas prior releases of FOCUS did not allow an index to span multiple concatenated files. Should you wish to concatenate the data sources comprising the index, you must issue the appropriate USE command prior to the REBUILD. The USE must include all cross-referenced and LOCATION files. REBUILD EXTERNAL INDEX contains an add function that allows you to append only new index records from a concatenated data source to the index data source, eliminating the need to recreate the index data source.

The original data source from which an index was built may not appear in the USE list when additional index records are added. If it does, REBUILD EXTERNAL INDEX will generate the following error message:

```
(FOC999) WARNING. EXTERNAL INDEX COMPONENT REUSED: ddname
```

## Positioning Indexed Fields

This feature enables you to create relationships between source and target segments, where the source segment is the segment containing the indexed field, and the target segment is any segment above or below the source segment within its path. You can position retrieval of indexed values within a particular segment in order to enhance retrieval performance by entering the hierarchy at a segment other than the root.

Positioning requires that the indexed field be a virtual field, not a real field. Since the DEFINE can contain any valid FOCUS expression, you can use real fields for positioning by assigning them temporary fieldnames and using those temporary field names for REBUILD EXTERNAL INDEX operations.

If the target segment is not in the same path as the source, the following error message will be generated:

```
(FOC974) EXTERNAL INDEX ERROR. INVALID TARGET SEGMENT
```

While a source segment can be a cross-referenced or location segment, target segments cannot be cross-referenced segments. If an attempt is made to place the target on a cross-referenced segment the following error message is generated:

```
(FOC1000) INVALID USE OF CROSS REFERENCE FIELD
```

If you do not choose to associate your index with a particular field, the source and target segments will be the same.

## Syntax

### How to Activate an External Index

After building an external index data source, you must associate it with the data sources from which it was created. You do this with the USE command. The syntax is the same as when you issued it prior to building the external index data source, but the WITH option is now available. The syntax is:

```
USE [ADD/REPLACE]
    database_name [AS mastername]
index_database_name {WITH/INDEX} mastername
.
.
END
```

where:

#### ADD

Appends one or more new data sources to the present USE list. Without the ADD option, the existing USE list is cleared and then replaced by the current USE list of data sources.

#### REPLACE

Replaces an existing *database\_name* in the USE list.

#### *database\_name*

Is the name of the data source. This is the ddname allocated for the data source in MVS, and the file name, file type, and file mode in CMS.

#### AS

Is used with a Master File name to concatenate data sources.

#### *mastername*

Specifies the FOCUS Master File name.

#### *index\_database\_name*

Is the name of the external index data source. In MVS, this is the ddname allocated to the index data source. In CMS, this is the file id (file name, file type, file mode) of the index data source.

#### WITH INDEX

Are new commands that create the relationship between the component data sources and the index data source. They can be used interchangeably.

## Example      Creating an External Index from a Concatenated Data Source

The following example illustrates how to create an external index from a concatenated data source.

```
USE CLEAR *
USE
EMPLOYEE
EMP2 AS EMPLOYEE
JOBFILE
EDUCFILE
END
```

Note that EMPLOYEE and EMP2 are concatenated and can be described by the EMPLOYEE Master File. To initiate REBUILD, enter the following at the FOCUS prompt:

```
REBUILD
```

You are prompted to select an option:

```
Enter option
```

1. REBUILD                    (Optimize the database structure)
2. REORG                     (Alter the database structure)
3. INDEX                     (Build/modify the database index)
4. EXTERNAL INDEX         (Build/modify an external index database)
5. CHECK                    (Check the database structure)
6. TIMESTAMP               (Change the database timestamp)
7. DATE NEW                (Convert old date formats to smartdate formats)
8. MDINDEX                 (Build/modify a multidimensional index. FUSION DBs only)
9. MIGRATE                 (Convert FOCUS masters/DBs to FUSION)

Select the external index option by entering:

```
EXTERNAL INDEX
```

REBUILD responds with:

```
NEW, OR ADD TO EXISTING INDEX DATABASE? (NEW/ADD)
```

For this example, assume you are creating a new index data source, and respond NEW. REBUILD then prompts:

```
ENTER FILENAME OF EXTERNAL INDEX
```

Enter the name of the external index data source, for example:

```
EMPIDX
```

You are prompted for the name of the data source from which the index information is obtained:

```
ENTER NAME OF FOCUS/FUSION FILE TO INDEX
```

Enter the name of the data source from which the index records are obtained:

```
EMPLOYEE
```

You are then prompted for the name of the field to index:

```
ENTER NAME OF FIELD TO INDEX
```

Enter the name of the field:

```
CURR_JOBCODE
```

You are prompted for a field to associate the index with. This allows for positioning of the indexed field outside the source segment:

```
ASSOCIATE INDEX WITH A PARTICULAR FIELD? (YES/NO)
```

Reply NO for this example.

Lastly, you are prompted for any record selection tests. Again, reply NO to the prompt:

```
ANY RECORD SELECTION TESTS? (YES/NO)
```

At this point the REBUILD process will continue and display the following when it has completed:

```
EXTERNAL INDEX FILE:          EMPIDX
DATE/TIME OF LAST CHANGE:    10/12/94 10.59.55
```

```
EXTERNAL INDEX DATABASE PAGES: 00000001
DATABASE INDEXED:             EMPLOYEE
FIELD NAME:                   CURR_JOBCODE
FIELD FORMAT:                 A3
SEGMENT NAME:                 EMPINFO
SEGMENT LOCATION:            EMPLOYEE
```

```
EXTERNAL INDEX DATA COMPONENTS:
```

```
EMPLOYEE
EMP2
```

This output is generated by the ? FDT command, which is automatically performed at the end of the REBUILD EXTERNAL INDEX process. This is done in order to validate the contents of the index data source.

The ? FDT command includes a display for the index data source.

Record selection may also be used by responding YES to the prompt:

```
ANY RECORD SELECTION TESTS? (YES/NO)
```

You are then prompted:

```
ENTER IF OR WHERE TESTS (TERMINATE WITH 'END' ON A SEPARATE LINE)
```

Selection tests may span more than one line and must be terminated by the END command. To select particular instances of CURR\_JOBCODE, you could respond:

```
IF CURR_JOBCODE EQ 'B02' OR 'B03' OR 'B04'  
OR 'B14'  
END
```

## Example

### Activating the External Index Data Source

You must activate the external index data source prior to referencing it in TABLE. This is done by placing it in the USE list as follows:

```
USE CLEAR  
USE  
EMPLOYEE FOCUS A  
EMP2 FOCUS A AS EMPLOYEE  
JOBFILE FOCUS A  
EDUCFILE FOCUS A  
EMPIDX FOCUS A WITH EMPLOYEE  
END  
JOIN CLEAR *  
JOIN JOBCODE IN JOBFILE TO CURR_JOBCODE IN EMPLOYEE AS AJ  
TABLE FILE JOBFILE  
PRINT FIRST_NAME LAST_NAME CURR_JOBCODE JOB_DESC  
IF JOBCODE EQ 'A07'  
END
```

External indexes do work on joined data structures so this request finds the following record meeting the specified criteria:

FIRST_NAME	LAST_NAME	CURR_JOBCODE	JOB_DESC
ALFRED	STEVENS	A07	SECRETARY

## Example

### Using an External Index to Join to Multiple Fields in a FOCUS Database

This example shows a way to effectively join to multiple fields in a FOCUS database, by using an external index built on a concatenated define file in CAR, COUNTRY | CAR called CARCOU.

```
FILE=FILE2,SUFFIX=FOCUS  
SEGNAME=CONT,SEGTYPE=S01  
FIELDNAME=CONTINENT,,A10,$  
SEGNAME=COUNTRY,SEGTYPE=S01,PARENT=CONT  
FIELDNAME=COUNTRY,,A10,$  
SEGNAME=CAR,SEGTYPE=S01,PARENT=COUNTRY  
FIELDNAME=CAR,,A16,$  
  
USE CAR FOCUS A AS CAR
```

```

CARCOU FOCUS A WITH CAR
END
JOIN CLEAR *
JOIN COUNTRY AND CAR IN FILE2 TO CARCOU IN CAR AS AJ
TABLE FILE FILE2
PRINT RCOST DCOST SALES
BY COUNTRY BY CAR
END

```

COUNTRY	CAR	RETAIL_COST	DEALER_COST	SALES
----	---	-----	-----	-----
ENGLAND	JAGUAR	8,878	7,427	0
		13,491	11,194	12000
	JENSEN	17,850	14,940	0
FRANCE	TRIUMPH	5,100	4,292	0
	PEUGEOT	5,610	4,631	0
ITALY	ALFA ROMEO	6,820	5,660	12400
		6,820	5,660	13000
		5,925	4,915	4800
	MASERATI	31,500	25,000	0
JAPAN	DATSUN	3,139	2,626	43000
	TOYOTA	3,339	2,886	35030
W GERMANY	AUDI	5,970	5,063	7800
	BMW	5,940	5,800	8950
		6,355	6,000	8900
		13,752	10,000	14000
		14,123	11,000	18940
W GERMANY		9,097	8,300	14000
	BMW	9,495	8,400	15600

## Adding a New Data Source to an Existing Index Data Source

Since EMP3 can be described by the EMPLOYEE Master File, you should clear the original USE list and reissue the USE command, concatenating EMP3 but omitting references to the EMPLOYEE and EMP2 data sources. Your USE must also point to EMPIDX, the index data source, in order for EMPIDX to be updated with information from EMP3. You can run REBUILD from a FOCEXEC, rather than interactively as follows. The FOCEXEC contains:

```

USE CLEAR
USE
EMP3 AS EMPLOYEE
JOBFILE
EDUCFILE
EMPIDX
END

```

```
REBUILD
EXTERNAL INDEX
ADD
EMPIDX
EMPLOYEE
CURR_JOBCODE
NO
NO
```

Now that the index data source has been updated with records from EMP3, you can issue the USE to activate the external index data source, and then execute your TABLE request.

The USE list references EMPLOYEE, EMP2, and EMP3, and EMPIDX uses the WITH option:

```
USE CLEAR
USE
EMPLOYEE
EMP2 AS EMPLOYEE
EMP3 AS EMPLOYEE
JOBFILE
EDUCFILE
EMPIDX WITH EMPLOYEE
END

TABLE FILE EMPLOYEE.CURR_JOBCODE
PRINT CURR_JOBCODE
IF CURR_JOBCODE EQ 'A07'
END
```

## Example

## Positioning an Indexed Field

The Master File for the file AUTOINS contains:

```
FILE=AUTOINS,SUFFIX=FOC,$
SEGNAME=STATE,SEGTYPE=S1,$
  FIELD=STATE_CODE,STATC,A2,$
  FIELD=STATE_COMPANY,STATCOMP,A10,$
SEGNAME=POLICY,PARENT=STATE,SEGTYPE=S1,$
  FIELD=AGENCY_NO,AGNO,A2,$
  FIELD=EXPIRE_DATE,EXPDTE,MDY,$
  FIELD=EFFECT_DATE,EFFDTE,MDY,$
  FIELD=HOMEOWNER,OWNER,A3,$
SEGNAME=VEHICLE,PARENT=POLICY,SEGTYPE=S1,$
  FIELD=VIN,VIN,A9,$
  FIELD=MODEL,MODEL,A15,$
  FIELD=MAKE,MAKE,A15,$
  FIELD=YEAR,YEAR,YY,$
SEGNAME=PRODUCTS,PARENT=VEHICLE,SEGTYPE=S1,$
  FIELD=PROD_CODE,PCODE,A2,$
  FIELD=COVERAGE,COVER,I8,$
  FIELD=CLASS_CODE,CCODE,A4,$
  FIELD=RATING,RATE,A3,$
```

A structure diagram follows:

```

STATE
01      S1
*****
*STATE_CODE **
*STATE_COMPA>**
*
*          **
*          **
*****
      I
      I
      I POLICY
02      I S1
*****
*AGENCY_NO   **
*EXPIRE_DATE **
*EFFECT_DATE **
*HOMEOWNER  **
*
*          **
*****
      I
      I
      I VEHICLE
03      I S1
*****
*VIN         **<---- Target
*MODEL      **
*MAKE       **
*YEAR       **
*
*          **
*****
      I
      I
      I PRODUCTS
04      I S1
*****
*PROD_CODE  **<---- Source
*COVERAGE   **
*CLASS_CODE **
*RATING     **
*
*          **
*****
*****

```

The PRODUCTS segment is the source segment, because the external index is built from a virtual field, SPCODE, whose value is based on PROD\_CODE. When you associate the index with the VEHICLE segment by choosing the VIN field in response to the prompt, ASSOCIATE INDEX WITH A PARTICULAR FIELD, the VEHICLE segment becomes the target segment. The following FOCEXEC creates this relationship:

```
DEFINE FILE AUTOINS
SPCODE/A2=PROD_CODE;
END
REBUILD
EXTERNAL INDEX
NEW
AUTOIDX
AUTOINS
SPCODE
YES
VIN
YES
WHERE SPCODE EQ 'A3'
END
```

The following is displayed by REBUILD when the process is complete:

```
EXTERNAL INDEX FILE:          AUTOIDX
DATE/TIME OF LAST CHANGE:    11/16/94 10.33.58

EXTERNAL INDEX FILE PAGES:    00000001
DATABASE INDEXED:            AUTOINS
FIELD NAME:                   SPCODE
FIELD FORMAT:                 A2
SEGMENT NAME:                 VEHICLE
SEGMENT LOCATION:            AUTOINS

EXTERNAL INDEX DATA COMPONENTS:

AUTOINS
```

## Special Considerations

- Up to eight indexes can be activated at a time in a USE list using the WITH command. More than eight indexes may be activated in a FOCUS session by issuing USE CLEAR and then issuing new USE commands.
- MODIFY may only make use of the external index using the FIND or LOOKUP functions. The external index cannot be used as an entry point, such as with MODIFY FILE filename.index.
- Indexes may not be created on field names longer than twelve characters.
- Text fields may not be used as indexed fields.
- The USE options NEW, READ, ON, LOCAL, and AS master on userid READ are not supported for the external index data source.
- The external index data source need not be allocated as CREATE FILE automatically does a temporary allocation. If a permanent data source is required then an allocation for the index data source ddname must be in place prior to the REBUILD EXTERNAL INDEX command.
- SORTOUT, the work file that the REBUILD EXTERNAL INDEX process creates, must be allocated with adequate space. The FOCUS default size is SPACE=(TRK,(5,5)), which may not be large enough to accommodate all retrieved index records. In order to estimate the space needed for SORTOUT, the following formula may be used:

`bytes = (field_length + 20) * number_of_occurrences`

## Error Messages Associated with External Indexes

The following error messages are associated with external indexes for FOCUS data sources.

- (FOC970)      **EXTERNAL INDEX SYNTAX ERROR: CONFLICTING OPTIONS**  
External index syntax does not support following USE command options:  
NEW, READ, ON, LOCAL, AS master ON userid READ.
- (FOC971)      **EXTERNAL INDEX SYNTAX ERROR: ILLEGAL DDNAME**  
The WITH command must be followed by a valid DDNAME, whose  
length is greater than 0 and less than or equal to 8 characters.
- (FOC972)      **EXTERNAL INDEX ERROR: MORE THAN 8 INDEXES IN USE**  
There can be maximum of 8 external Indexes in USE.
- (FOC973)      **EXTERNAL INDEX INVALID CONFIGURATION:**  
Not all components used to build concatenated external index are in use at  
this time, it is not safe to use this index.

- (FOC974)      **EXTERNAL INDEX ERROR: INVALID TARGET SEGMENT**  
Target segment for external index is invalid. Real data source fields cannot have a target segment different from the location segment.
- (FOC975)      **EXTERNAL INDEX ERROR: SEGMENT NOT IN PATH**
- (FOC976)      **EXTERNAL INDEX EXPIRED:**  
One of the component data sources was updated after the index was generated. It is not safe to use index under these conditions.
- (FOC977)      **EXTERNAL INDEXES CANNOT BE CREATED FOR NON-FOCUS DATABASES**  
External indexes may only be created for data sources with SUFFIX=FOC.
- (FOC999)      **WARNING. EXTERNAL INDEX COMPONENT REUSED: ddname**  
Attempt is made to add data to an external index from a data source that was previously used as a component.
- (FOC1000)     **INVALID USE OF CROSS REFERENCE FIELD**  
Cross-reference fields may only be used as an external index if they are associated with a real data source field. An external index cannot be associated with a cross-reference field.

## Using Indexed Fields for Retrieval Automatically (AUTOINDEX)

FOCUS automatically expedites data retrievals by exploiting indexed fields in (most) cases where TABLE requests contain equality or range tests on those fields.

Requests must contain equality or range tests against indexed fields in the highest referenced segment in order for indexed retrievals to be performed with AUTOINDEX=ON. While equality and range tests are permitted on all data types, range tests on packed data will force FOCUS to perform sequential retrieval.

### Syntax

#### Setting AUTOINDEX on or off

AUTOINDEX is on by default. Use the SET command syntax below to turn it off or back on:

```
SET AUTOINDEX = {ON|OFF}
```

where:

ON

Uses indexed retrieval when possible. ON is the default.

OFF

Sets AUTOINDEX off. When AUTOINDEX is OFF, indexed retrievals are only performed when explicitly specified using indexed views.

## Example

## Using AUTOINDEX

Consider the following Master File:

```
FILENAME=INVENT , SUFFIX=FOC ,
  SEGNAME=STOR_SEG , SEGTYPE=S1 ,
    FIELDNAME=AREA , ALIAS=LOC , FORMAT=A1 , $
  SEGNAME=DATE_SEG , PARENT=STOR_SEG , SEGTYPE=SH1 ,
    FIELDNAME=DATE , ALIAS=DTE , FORMAT=A4MD , $
  SEGNAME=DEPT , PARENT=DATE_SEG , SEGTYPE=S1 ,
    FIELDNAME=DEPARTMENT , ALIAS=DEPT , FORMAT=A5 , FIELDTYPE=I , $
    FIELDNAME=DEPT_CODE , ALIAS=DCODE , FORMAT=A3 , FIELDTYPE=I , $
    FIELDNAME=PROD_TYPE , ALIAS=PTYPE , FORMAT=A10 , FIELDTYPE=I , $
  SEGNAME=INVENTORY , PARENT=DEPT , SEGTYPE=S1 , $
    FIELDNAME=PROD_CODE , ALIAS=PCODE , FORMAT=A3 , FIELDTYPE=I , $
    FIELDNAME=UNIT_SOLD , ALIAS=SOLD , FORMAT=I5 , $
    FIELDNAME=RETAIL_PRICE , ALIAS=RP , FORMAT=D5.2M , $
    FIELDNAME=DELIVER_AMT , ALIAS=SHIP , FORMAT=I5 , $
```

The following FOCEXEC contains an equality test on DEPT\_CODE and PROD\_CODE. DEPT\_CODE is used for indexed retrieval since it is in the higher referenced segment.

```
SET AUTOINDEX=ON
TABLE FILE INVENT
SUM UNIT_SOLD RETAIL_PRICE
IF DEPT_CODE EQ 'H01'
IF PROD_CODE EQ 'B10'
END
```

If a TABLE request contains an equality or range test against more than one indexed field in the same segment, AUTOINDEX uses the first index referenced in that segment for retrieval. The following FOCEXEC contains an equality test against two indexed fields. Since DEPT\_CODE occurs before PROD\_TYPE in the Master File, DEPT\_CODE is used for retrieval:

```
SET AUTOINDEX=ON
TABLE FILE INVENT
SUM UNIT_SOLD AND RETAIL_PRICE
IF PROD_TYPE EQ 'STEREO'
IF DEPT_CODE EQ 'H01'
END
```

If the equality or range test is against an indexed field that does not reside in the highest referenced segment, indexed retrieval does not occur. In the following example, indexed retrieval is not performed because the request contains a reference to AREA, a field in the STOR\_SEG segment:

```
SET AUTOINDEX=ON
TABLE FILE INVENT
SUM UNIT_SOLD AND RETAIL_PRICE
BY AREA
IF PROD_CODE EQ 'B10'
IF PROD_TYPE EQ 'STEREO'
END
```

**Reference**

**Implications with Requests Containing Indexed Views**

When indexed retrieval is explicitly specified through an indexed view (as in TABLE FILE filename.indexed\_fieldname) indexed retrievals are performed in the following situations:

- When AUTOINDEX is set to OFF and a request contains an equality test on the indexed field.
- When AUTOINDEX is set to ON and a request contains either an equality or a range (FROM ... TO) test against the indexed field.

**Reference**

**Special Considerations for AUTOINDEX**

- AUTOINDEX is never used when a TABLE request contains alternate file views (for example, TABLE FILE filename.fieldname).
- Indexed retrievals are not performed when TABLE requests contain BY HIGHEST or BY LOWEST phrases and AUTOINDEX is ON. They are only performed when the request uses an indexed view.
- When using external indexes, you must explicitly code for indexed retrieval by using the syntax TABLE FILE filename.indexed\_fieldname, as AUTOINDEX does not automatically utilize external indexes.

# Using External Sorts to Expedite Large Reports

Expedite the processing of large reports by invoking external sorts, such as DFSORT or SyncSort. Gains achieved by HiperFOCUS customers led us to make external sorts available to all FOCUS for S/390 customers. While the impact varies from case to case, external sorts significantly decrease CPU time in most situations where many lines of report output are produced. You can call for external sorts in any TABLE, EMR, MATCH or GRAPH request.

## Syntax

### Controlling External Sort Operations

Enable external sorts by issuing the SET EXTSORT command:

```
SET EXTSORT = {ON|OFF}
```

where:

ON

Directs FOCUS to use an external product for sorting reports. This is the default.

OFF

Selects the FOCUS internal sort procedure for sorting reports.

To verify the sort used, issue ? STAT after the report request and examine the SORT USED parameter:

FOCUS	The internal FOCUS sort was used.
SQL	RDBMS sort facilities (accessed by FOCUS) used to sort report.
EXTERNAL	An external product was used to sort the report. To confirm the extent of external sorting, issue ? STAT. The INTERNAL MATRIX CREATED parameter will display NO if an external sort handled the report and YES if FOCUS handled the first group of records and an external sort handled the remainder.
NONE	The report did not require sorting.

## External Sort Products Supported

The following sort utilities are supported:

- MVS - DFSORT, SyncSort, PLSORT
- CMS - DFSORT, SyncSort, VMSORT (must be 31-bit addressable).

### CMS External Sort Requirements

In CMS, issue a GLOBAL TXTLIB command to identify the location of the sort software. If FOCUS cannot locate the sort, it will issue a GLOBAL TXTLIB SORTLIB command. If FOCUS still cannot locate the sort software, it terminates with the following error message:

```
(FOC909) CRITICAL ERROR IN EXTERNAL SORT. RETURN CODE IS: 16
```

You also receive an error message in CMS if your system sort is DFSORT/CMS and your temp disk is not large enough for your sort requirements. With DFSORT/CMS, the maximum temp disk space obtainable for a 3380 device is 16 cylinders. If this is not large enough for your sort requests, you may have to modify the DUFMAC MACRO to allocate more space for sort work files.

FOCUS always respects the installation parameters of your sort product.

## Displaying External Sort Messages

By default, FOCUS does not display the messages produced by external sort products. It is possible, however, to display them for diagnostic purposes. See instructions below for displaying DFSORT and SyncSort messages under MVS.

### Procedure

#### Displaying DFSORT messages

1. Create a sequential file with these attributes:

```
DCB=(LRECL=80,RECFM=F,BLKSIZE=80)
```

2. Create the following record beginning in column 2:

```
OPTION MSGPRT=ALL, MSGDDN=trace_name
```

where:

```
trace_name
```

Is any user-defined ddname.

3. Allocate the file to ddname DFSPARM.
4. Allocate the ddnames *trace\_name* and SORTDIAG to a single file or to SYSOUT. This ensures that all trace and diagnostic messages are written to the same file.

**Procedure**      **Displaying SyncSort Messages**

1. Create a file with these attributes:

```
DCB=(LRECL=80,RECFM=F,BLKSIZE=80)
```

2. Create the following record beginning in column 2:

```
MSG=AB,BMSG,MSGDD=trace_name,LIST
```

where:

```
trace_name
```

Is any user-defined ddname.

3. Allocate the file to ddname \$ORTPARM.
4. Allocate the ddname *trace\_name* to a file or to SYSOUT.

## Using External Sorts to Optimize Aggregation

You can significantly improve performance when sorting and aggregating records by passing off the aggregation operations to your external sort product as well as the sorting, rather than having FOCUS do them. The gains are particularly notable when issuing relatively simple requests against large databases.

**Syntax**      **Using Aggregation in Your External Sort**

```
SET EXTAGGR = operation
```

where:

```
operation
```

Is one of the following:

- OFF**      Disallows aggregation by an external sort.
- NOFLOAT**      Allows aggregation if there are no floating data fields present.
- ON**      Allows aggregation by an external sort. ON is the default.

**Reference**      **Conditions for Aggregating with an External Sort**

- You must be using SYNCSORT or DFSORT.
- EXTAGGR cannot be set to OFF.
- Queries should be simple. (AUTOTABLEFable)
- The PRINT display command may not be used in the query.
- SET ALL must be equal to OFF.

- Only the following column prefixes are allowed: SUM, AVG, CNT, FST.
- Columns can be computed or have row-totals.
- CMS DFSORT does not support aggregation of numeric data types. When SET EXTAGGR = NOFLOAT and your query aggregates numeric data, the external sort is not called and aggregation is performed by the FOCUS sort.

## Example

### How External Sorts Can Change the Sequence of Output

Using an external sort for aggregation may change the record sequence in a report. For example, if you use SUM on an alphanumeric field in your report request without using an external sort, FOCUS displays the last instance of the sorted fields in the output. Turning on aggregation in the external sort results in the first record being displayed.

With aggregation in the external sort turned on:

```
SET EXTAGGR = ON
TABLE FILE CAR
SUM CAR BY COUNTRY
END
```

The output is:

```
COUNTRY      CAR
-----      ---
ENGLAND      JAGUAR
FRANCE       PEUGEOT
ITALY        ALFA ROMEO
JAPAN        DATSUN
W GERMANY    AUDI
```

With external aggregation turned off:

```
SET EXTAGGR = OFF
TABLE FILE CAR
SUM CAR BY COUNTRY
END
```

The output is:

```
COUNTRY      CAR
-----      ---
ENGLAND      TRIUMPH
FRANCE       PEUGEOT
ITALY        MASERATI
JAPAN        TOYOTA
W GERMANY    BMW
```

#### Tip:

Use of SET SUMPREFIX in conjunction with external aggregation also affects the order of information displayed in your report.

## Reference

### Behavioral Change With External Aggregation

When aggregation is performed by an external sort the statistical variables &RECORDS and &LINES will contain equal values. This is because the external sort products do not return line counts for the answer sets. This behavioral change could affect existing code that checks for a value of &LINES.

## Using External Sorts to Expedite Production of HOLD Files

You can save up to twenty percent on processing time by using an external sort to generate HOLD files. This is especially effective when making relatively simple requests against large databases.

## Syntax

### Creating HOLD Files with an External Sort

```
SET EXTHOLD = [OFF|ON]
```

where:

OFF

Disables HOLD files by an external sort. OFF is the default.

ON

Enables HOLD files by an external sort.

## Conditions for Use

- The EXTSORT=ON default must be in effect.
- EXTHOLD must be ON.
- The request must have a BY field.
- The request must contain ON TABLE HOLD or ON TABLE HOLD AS.
- Use for simple queries (such as those for which AUTOTABLEF is used).

Note: AUTOTABLEF analyzes queries and determines whether the requested operations and formatting options require the internal matrix or not. If a matrix is not needed to satisfy the query, FOCUS avoids the overhead associated with creating a matrix. The internal matrix is stored in a file or data set named FOCSORT. The FOCSORT default is ON to maximize potential performance gains.

- SET ALL must be OFF.
- IF/WHERE TOTAL or BY TOTAL cannot be included in the request.

- EXTAGGR must be ON for requests containing SUM, and SUM and FST are the only column prefixes allowed.
- For requests containing PRINT commands, the only column prefixes allowed are SUM, AVE, MAX, MIN, FST and LST.

### Example

### Generating a Small HOLD File from a Large Database with an External Sort

This request efficiently produces a small extract from the EMPLOYEE database:

```
SET EXTSORT=ON
SET EXTHOLD=ON
SET NODATA= ' '
DEFINE FILE EMPLOYEE
TGROSS/P11.2CS=GROSS;
END
TABLE FILE EMPLOYEE
PRINT TYPE DAT_INC PAY_DATE TGROSS
BY EMP_ID
BY BANK_NAME
ON TABLE HOLD AS LOADEMP
END
TABLE FILE LOADEMP
PRINT *
END
```

Here are the results of this request:

PAGE 1

EMP_ID	BANK_NAME	TYPE	DAT_INC	PAY_DATE	TGROSS
-----	-----	----	-----	-----	-----
071382660		BANK	82/01/01	82/08/31	916.67
071382660			81/01/01	82/07/30	916.67
071382660				82/06/30	916.67
071382660				82/05/28	916.67
071382660				82/04/30	916.67
071382660				82/03/31	916.67
071382660				82/02/26	916.67
071382660				82/01/29	916.67
071382660				81/12/31	833.33
071382660				81/11/30	833.33
112847612		BANK	82/01/01	82/08/31	1,100.00
112847612				82/07/30	1,100.00
112847612				82/06/30	1,100.00
112847612				82/05/28	1,100.00
112847612				82/04/30	1,100.00
112847612				82/03/31	1,100.00
112847612				82/02/26	1,100.00

MORE

PAGE 2

EMP_ID	BANK_NAME	TYPE	DAT_INC	PAY_DATE	TGROSS
-----	-----	----	-----	-----	-----
112847612				82/01/29	1,100.00
117593129	STATE	BANK	82/06/01	82/08/31	1,540.00
117593129	STATE	HSM	82/05/01	82/07/30	1,540.00
117593129	STATE			82/06/30	1,540.00
117593129	STATE			82/05/28	1,479.50
119265415		BANK	82/05/14	82/08/31	791.67
119265415		HSM	82/01/04	82/07/30	791.67
119265415				82/06/30	791.67
119265415				82/05/28	791.67
119265415				82/04/30	754.17
119265415				82/03/31	754.17
119265415				82/02/26	754.17
119265415				82/01/29	754.17
119329144	BEST BANK	HSM	82/08/01	82/08/31	2,475.00
123764317	ASSOCIATED	BANK	82/05/14	82/08/31	2,238.50
123764317	ASSOCIATED	HSM	82/01/04	82/07/30	2,238.50
123764317	ASSOCIATED			82/06/30	2,238.50

MORE

Expediting Retrievals and Processing

---

PAGE 3

EMP_ID	BANK_NAME	TYPE	DAT_INC	PAY_DATE	TGROSS
-----	-----	---	-----	-----	-----
123764317	ASSOCIATED			82/05/28	2,238.50
123764317	ASSOCIATED			82/04/30	2,035.00
123764317	ASSOCIATED			82/03/31	2,035.00
123764317	ASSOCIATED			82/02/26	2,035.00
123764317	ASSOCIATED			82/01/29	2,035.00
126724188		BANK	82/07/01	82/08/31	1,760.00
126724188		HSM		82/07/30	1,760.00
126724188				82/06/30	1,760.00
126724188				82/05/28	1,760.00
219984371		BANK	82/01/01	82/08/31	1,540.00
326179357	ASSOCIATED	BANK	82/04/01	82/08/31	1,815.00
326179357	ASSOCIATED	HM		82/07/30	1,815.00
326179357	ASSOCIATED	SICK		82/06/30	1,815.00
326179357	ASSOCIATED			82/05/28	1,815.00
326179357	ASSOCIATED			82/04/30	1,815.00
451123478	ASSOCIATED	BANK	82/05/14	82/08/31	1,342.00
451123478	ASSOCIATED	HSM	82/02/02	82/07/30	1,342.00

MORE

PAGE 4

EMP_ID	BANK_NAME	TYPE	DAT_INC	PAY_DATE	TGROSS
-----	-----	---	-----	-----	-----
451123478	ASSOCIATED			82/06/30	1,342.00
451123478	ASSOCIATED			82/05/28	1,342.00
451123478	ASSOCIATED			82/04/30	1,254.00
451123478	ASSOCIATED			82/03/31	1,254.00
451123478	ASSOCIATED			82/02/26	1,254.00
543729165		BANK	82/06/11	82/08/31	750.00
543729165		HSM	82/04/01	82/07/30	750.00
543729165				82/06/30	750.00
543729165				82/04/30	720.84
818692173	BANK ASSOCIATION	BANK	82/04/09	82/08/31	2,235.00
818692173	BANK ASSOCIATION	HSM	81/11/02	82/07/30	2,235.00
818692173	BANK ASSOCIATION			82/06/30	2,235.00
818692173	BANK ASSOCIATION			82/05/28	2,235.00
818692173	BANK ASSOCIATION			82/04/30	2,235.00
818692173	BANK ASSOCIATION			82/03/31	2,147.75
818692173	BANK ASSOCIATION			82/02/26	2,147.75
818692173	BANK ASSOCIATION			82/01/29	2,147.75

MORE

PAGE 5

EMP_ID	BANK_NAME	TYPE	DAT_INC	PAY_DATE	TGROSS
-----	-----	----	-----	-----	-----
818692173	BANK ASSOCIATION			81/12/31	2,147.75
818692173	BANK ASSOCIATION			81/11/30	2,147.75

---

## CHAPTER 3

# Exploiting System Enhancements and Raised Limits

### Topics:

- Improving Page Handling With TRACKIO
- Reducing I/O With MINIO Database Access Method
- Employing the IBI MVS Subsystem

This chapter introduces system enhancements for both simplifying and expanding your operations and for lowering your operating costs.

## Improving Page Handling With TRACKIO

The latest OS/390 and MVS releases of FOCUS provide improved page handling, significantly reducing I/O requirements and elapsed times experienced when accessing FOCUS files. With the TRACKIO feature on (the current default) MVS gathers more pages to fill a track before reading or writing the pages to disk.

### **Syntax**    Turning TRACKIO On or Off

TRACKIO is on by default and you use the following SET command to change its setting:

```
SET TRACKIO = {ON|OFF}
```

where:

ON

Causes FOCUS to fill a track before reading or writing to disk. (the default)

OFF

Uses the old method for TRACKIO.

To determine whether TRACKIO is ON, just issue ? SET at the FOCUS prompt.

To view performance gains achieved with TRACKIO, simply run requests with TRACKIO ON and then OFF and compare the relative EXCP statistics.

## Reducing I/O With MINIO Database Access Method

The MINIO buffering technique greatly improves FOCUS performance under MVS by reducing the number of I/O operations performed when accessing FOCUS data sources. With MINIO on, no block is ever read more than once, so the reads equal the number of tracks. This reduces elapsed times when reading and writing.

When used with FOCUS data sources that are not disorganized, MINIO can reduce data source I/O operations for TABLE and MODIFY commands by up to fifty percent. The actual reductions vary depending on data source structures and average number of children per parent segment. By reducing I/O, elapsed times for TABLE and MODIFY commands also drop.

### Syntax Using MINIO to Reduce I/Os

```
SET MINIO = {ON|OFF}
```

where:

ON

Enables MINIO. ON is the default.

OFF

Disables MINIO.

### MINIO Usage

MINIO also reduces CPU time slightly while slightly increasing memory utilization. MINIO requires a track I/O buffer per referenced segment type. Between 40 and 48K of above-the-line virtual memory is needed per referenced segment, depending on the data device type.

With MINIO enabled, FOCUS bases its decision on whether or not to employ it on the command, as well as which data sources to use it with. In executing a single command that references several data sources, MINIO might be used for some but not for others. Data sources accessed through indexes, or those physically disorganized through online updates where target records are badly out of sequence, are not candidates for MINIO buffering. When this occurs, FOCUS abandons MINIO buffering techniques and applies standard I/O methodology.

Use MINIO for reading data with TABLE, TABLEF, GRAPH, MATCH or for the DUMP phase of the REBUILD command, provided the target data source is not physically disorganized or accessed through an index.

Use MINIO with MODIFY (not MAINTAIN), provided no CRTFORM or COMMIT subcommands are present. (CRTFORMs indicate online transaction processing, which requires that completed transactions be written out to the data source. COMMITs are explicit orders to do so. Both are incompatible with MINIO minimization logic, ruling out its use.) The use of MINIO with MODIFY also requires that the data source be accessed sequentially. Any attempt to access an index, or update a physically disorganized data source disables MINIO. In addition, frequent repositioning to previously accessed records, even within well-organized data sources disables MINIO.

Use the ? STAT command to determine whether the previous data source access command employed MINIO. It generates a screen similar to the following:

```

                                STATISTICS OF LAST COMMAND
RECORDS          =           0      SEGS CHNGD      =           0
LINES            =           0      SEGS DELTD      =           0
BASEIO          =           87      NOMATCH        =           0
TRACKIO         =           16      DUPLICATES     =           0
SORTIO          =           0      FORMAT ERRORS  =           0
SORT PAGES      =           0      INVALID CONDTs =           0
READS           =           1      OTHER REJECTS  =           0
TRANSACTIONS    =          1500     CACHE READS    =           0
ACCEPTED        =          1500     MERGES         =           0
SEGS INPUT      =          1500     SORT STRINGS   =           0

INTERNAL MATRIX CREATED: YES      AUTOINDEX USED:      NO
SORT USED:          FOCUS         AUTOPATH USED:      NO
MINIO USED:          YES
```

In this example, MINIO USED displays YES. The other alternatives are NO or DISABLED.

- YES means MINIO buffering took place, reducing the number of tracks read/written to the FOCUS data source.
- NO, means MINIO buffering did not take place.
- DISABLED means MINIO buffering was started but terminated, as no performance gains were possible. This does not indicate unsuccessful completion, only that MINIO buffering started and ended during the read/write operation.

## Reference MINIO Usage Restrictions/Considerations

- When MINIO is used with MODIFY, all CHECK subcommands are ignored. If a MODIFY command terminates abnormally, the condition of the data source will be unpredictable, and should therefore be restored from a backup copy and the update repeated. Since MINIO is designed for minimizing I/O during large scale data source loads and updates, there is no checkpoint or restart facility on the assumption that the whole operation will be repeated. If this situation is unacceptable, set MINIO off.
- MINIO cannot be used to access data sources through a Central Database Server (formerly *sink machines*) or an HLI program.
- MINIO requires the TRACKIO feature (TRACKIO must be ON—the default setting). If TRACKIO is OFF, MINIO is deactivated.
- MINIO buffering starts when a FOCUS data source exceeds 64 pages in size. MINIO is not activated for smaller data sources.
- When files are modified through UPDATE, INCLUDE or DELETE operations on indexed fields, MINIO is disabled (if FIELDTYPE=I or INDEX=I appears in the Master File for the field).
- CRTFORM and COMMIT commands disable MINIO.
- MAINTAIN procedures cannot use MINIO buffering techniques.
- MINIO is disabled when target data sources become physically disorganized through transaction processing.

## Employing the IBI MVS Subsystem

The IBI Subsystem, which replaces the IBI SVC, provides state-of-the-art communication and coordination services between the MVS address spaces where Information Builders products run. In addition to improved ease of use and additional functionality, the IBI Subsystem also simplifies installation and maintenance procedures.

The Subsystem expedites installation of both FOCUS and MSO. In addition to improving performance and supporting higher capacities, it also delivers enhanced error-recovery capabilities. It provides the underlying communications support for MSO, Simultaneous Usage (SU), SmartMode, Application Control Environment (ACE), and IMS BMP, and the HiperFOCUS HiperBudget feature employs it to regulate system-wide use of expanded storage.

For additional information about the Subsystem, see the new *OS/390 and MVS Installation Guide, Version 7.1 (DNI000994.1100)*.

# Index

## Numeric index entries

2-gigabyte databases, 2-5

## A

Access File, 2-6  
  attributes, 2-8  
  introduction, 2-7  
  syntax, 2-9

Access File attributes  
  DATANAME, 2-9  
  LOCATION, 2-10  
  MASTERNAME, 2-9  
  WHERE, 2-9

ACCESSFILE attribute, 2-6

aggregating and sorting columns in one pass, 1-20

aggregations  
  conditions for use of external sorts, 2-43  
  optimizing with external sorts, 2-43

attributes  
  for FOCUS Access Files, 2-8

AUTOINDEX, 2-38

AUTOPATH  
  optimizing retrieval paths automatically, 2-15

AUTOPATH restrictions  
  LAST and some self-referencing fields, 2-15

## B

boundaries for Pooled Tables, 1-7

BY TOTAL feature, 1-20

## C

clusters, 1-2

column aggregation and sorting in one pass, 1-20

concatenating unlike files, 2-17

## D

DATANAME attribute, 2-9

DFSORT, 1-4

## E

estimating records to be pooled, 1-3

estimating report lines. *See* ESTLINES

ESTLINES, 1-3

ESTRECORDS, 1-3

External Index feature, 2-25

external sorts  
  expediting large reports, 2-41  
  expediting production of HOLD files, 2-45  
  optimizing aggregations, 2-43  
  products supported, 2-42

## F

FIXRET. *See* also FIXRETRIEVE

FOC2GIGDB, 2-5

FOCPARM  
  configuring Pooled Tables parameters, 1-19

FOCPOOLT, 1-12

FOCUS database  
  Access File, 2-6  
  partitioned, 2-5

FOCUS database partitions

intelligent, 2-5

## H

HOLD files

produced efficiently with external sorts, 2-45

## I

I/O buffering, 3-3

IBI Subsystem, 3-6

installation

Pooled Tables, 1-18

installing Pooled Tables, 1-18

installing Pooled Tables on MVS, 1-18

installing Pooled Tables on VM/CMS, 1-18

intelligent partitioning, 2-5

## J

joins

and Access Files, 2-12

and partitioned FOCUS data sources, 2-12

## K

keyed retrievals from HOLD files, 2-2

## L

LOCATION attribute

in Access File, 2-10

## M

Master File

ACCESSFILE attribute, 2-6

MASTERNAME attribute, 2-9

MAXEXTSRSTS, 1-4

MINIO, 3-3

considerations for use, 3-5

restrictions, 3-5

MORE command, 2-17. *See* Universal

Concatenation

concatenating unlike files, 2-17

MVSMSGDF, 1-4

MVSMSGSS, 1-4

## O

optimizing retrieval paths automatically

AUTOPATH, 2-15

options

Pooled Tables, 1-2

## P

page handling. *See* TRACKIO

partitions

and joins, 2-12

FOCUS database, 2-5

POOL, 1-2

POOLBATCH, 1-4

Pooled Tables

commands and sub pool boundaries, 1-7

common selection criteria, 1-13

configuration, 1-19

example, 1-5

FOCPARM parameters, 1-19

FOCPOOLT, 1-12

installing on all systems, 1-18

installing on MVS, 1-18

installing on VM/CMS, 1-18

memory management, 1-12

memory requirements, 1-12

single TABLE clusters, 1-8

sort selection, 1-15

statistics, 1-10

sub pool boundaries, 1-7

subroutines for use with, 1-9

trace facility, 1-15

use with batch requests, 1-14  
 use with non-relational databases, 1-13

#### Pooled Tables - continued

use with relational databases, 1-13

Pooled Tables temporary work file, 1-12

POOLFEATURE, 1-19

#### pooling

criteria, 1-2

POOLMEMORY, 1-3

POOLRESERVE, 1-3

positioning indexed fields, 2-26

## R

reduce I/Os retrieving from HOLD files, 2-2

reducing I/O, 3-3

## S

### SET

AUTOINDEX=ON/OFF, 2-38

AUTOPATH=ON/OFF, 2-15

ESTLINES, 1-3

ESTRECORDS, 1-3

EXTHOLD=OFF/ON, 2-45

FIXRET=ON/OFF, 2-2

FOC2GIGDB=ON/OFF, 2-5

MAXEXTSRSTS, 1-4

MINIO=ON/OFF, 3-3

POOL, 1-2

POOLBATCH, 1-4

POOLFEATURE, 1-19

POOLMEMORY, 1-3

POOLRESERVE, 1-3

SORTLIB, 1-4

TRACEOFF, 1-5

TRACEON, 1-4

TRACKIO=ON/OFF, 3-2

sort selection with Pooled Tables, 1-15

SORTLIB, 1-4

#### start pooling

SET POOL=ON, 1-2

#### starting the trace

SET TRACEON, 1-15

#### stop pooling

SET POOL=OFF, 1-2

#### sub pool

boundaries for Pooled Tables, 1-7

sub pool boundaries, 1-2

subroutines and Pooled Tables, 1-9

Subsystem. *See* IBI Subsystem

SYNCSORT, 1-4

## T

trace facility for Pooled Tables, 1-15

TRACEOFF, 1-5

TRACEON, 1-4

TRACKIO, 3-2

## U

Universal Concatenation, 2-17

## V

VMSORT, 1-4

## W

WHERE test

in Access File, 2-9

---

# Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. Send comments to:

Corporate Publications  
Attn: Manager of Documentation Services  
Information Builders  
Two Penn Plaza  
New York, NY 10121-2898

or FAX this page to (212) 967-0460, or call **James Patrick** at (212) 736-4433, x**3708**.

Name: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

Telephone: \_\_\_\_\_ Date: \_\_\_\_\_

Comments:

---

# Reader Comments