

Two-Gigabyte FOCUS Database Implementation Guide

A simplified guide to the new 2-GB FOCUS database for FOCUS customers

In FOCUS Version 7.1, the size of a FOCUS database file has been increased to a maximum of two gigabytes. The 2-GB FOCUS database overcomes previous FOCUS database size limitations. When used in conjunction with FOCUS's new partitioning capabilities, one FOCUS database can now span up to 500 gigabytes, while the prior FOCUS database size is limited to 64 gigabytes.

To enable support for FOCUS databases larger than one-gigabyte, you must set the FOC2GIGDB parameter to ON in the system-wide profile, FOCPARM.

Increased Storage Capacity

The most obvious application for the new FOCUS database is one where the current FOCUS database has reached its storage capacity. The maximum size of an older physical FOCUS database is one gigabyte (256K 4K pages). Through the use of LOCATION files, each segment of the database can be stored as a separate physical file allowing a theoretical maximum size of 64 gigabytes. This theoretical maximum is rarely reached because the lowest segments of a hierarchical file structure tend to include far more data than the parent segments in the hierarchy. Consequently, while the grandchild and great-grandchild segments may approach the size limit, the upper level segments are usually nowhere near capacity. Also, by applying the USE command for database concatenation, you can increase the logical size of a FOCUS database, but in doing so you lose the capability to perform indexed reads, so there are major performance considerations to weigh when concatenating FOCUS files.

Partitioning

The new FOCUS database is designed to support up to 250 physical database partitions, each up to two gigabytes, for a maximum of 500 gigabytes of real storage per logical database. Because this horizontal partitioning takes slices of the entire file structure, the database partitions are not subject to the multiplicative effect of LOCATION-style partitioning. The databases can grow in size over time, and can be repartitioned based on the requirements of the application. The database also supports vertical partitioning — a separate physical partition comprised of all instances of a single field — for even greater performance advantages and flexibility. Converting data from older FOCUS databases to the new FOCUS database partitioning scheme is a simple procedure, but it does take some analysis and planning to target the applications that will benefit most from the new capabilities.

Note: You do not have to partition your database.

Intelligent Partitioning

In addition to allowing up to 250 separate two-gigabyte files, the database supports intelligent partitioning. Intelligent partitioning places selection criteria on a partition. The data contained within the partition conforms to the selection criteria. With this information, FOCUS optimizes database access by retrieving only those partitions whose values are consistent with the selection criteria in the request. For example, a toy company records sales of the toys they manufacture. With the new FOCUS database, the toy sales can be partitioned by month: January sales are loaded into the January file; February sales are loaded into the February file; and so on. These partitions are identified in an Access File for the FOCUS database. To make these partitions intelligent, WHERE criteria are added to the partition declarations that are specified in the Access File to signal to the product that the data within each partition conforms to its WHERE criteria. See *The FOCUS Access File* on page 4 for more information.

Example Using Intelligent Partitioning

The following Access File illustrates how intelligent partitioning is used:

```
MASTERNAME TOYSALES
  DATANAME TOYSALES.JANUARY.FOCUS
    WHERE MONTH EQ 'JANUARY';
  DATANAME TOYSALES.FEBRUARY.FOCUS
    WHERE MONTH EQ 'FEBRUARY';
  .
  .
  .
  DATANAME TOYSALES.DECEMBER.FOCUS
    WHERE MONTH EQ 'DECEMBER';
```

Identifying Applications That Can Use Partitioning

To select applications that can benefit most from partitioning, look for applications that employ USE commands to concatenate databases or data that lends itself to intelligent partitioning, such as data stored by month or by department. Intelligent partitioning functions like an intelligent USE. It looks at the Access File to determine which partition to read, whereas the USE command reads all of the files specified. This intelligence decreases I/O operations and delivers significant performance benefits.

Implementing Partitions on an Existing FOCUS Database

Once you have targeted an application that can make use of partitioning, you must convert from the older FOCUS database to the new FOCUS database:

- 1. Add the ACCESS or ACCESSFILE attribute to the Master File**

Since partitioning requires an Access File, this file must be identified in the Master File using the ACCESS or ACCESSFILE attribute in the file declaration. The ACCESS or ACCESSFILE attribute tells the FOCUS database the physical name of the Access File where the partitioning information is stored.

- 2. Create a FOCUS Access File**

Create a FOCUS Access File for the new FOCUS database using any text editor. For each database file in the current FOCUS database's USE command, create a DATANAME entry in the Access File. If these database files contain data based on selection criteria — for instance month — code WHERE criteria under each partition that describes the selection criteria on which that data is based.

Separating an Existing Database File Into Intelligent Partitions

If the data is not separated into files that lend themselves to intelligent partitioning, use the FOCUS language to accomplish this.

- 1.** Issue TABLE requests with WHERE criteria and the ON TABLE HOLD phrase to create extracts of the data based on selection criteria.
- 2.** Use MODIFY to load each of these extracts into its own separate physical data file keeping track of the file name and selection criteria corresponding to the data the file contains.
- 3.** Create a FOCUS Access File for the new FOCUS database using any text editor. For each database file created in the prior step, create a DATANAME entry in the Access File with WHERE criteria following each DATANAME entry that describe the data.

The FOCUS Access File

The Access File provides comprehensive metadata management for all FOCUS files. It 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 to FOCUS must supply the name of a Master File. FOCUS reads the Master File and uses the declarations in it to access the database. If the Master File includes an ACCESS or ACCESSFILE attribute, FOCUS reads the named Access File and uses it to locate the correct data files. Each Master File can point to its own separate Access File, or several Master Files can point to the same 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 ACCESS or ACCESSFILE attribute, FOCUS attempts to satisfy the request with the Master File alone.

You can use an Access File to take advantage of the following database features:

- Horizontal partitioning. A database can consist of several separate files, or partitions, each of which contains the database records for a specific time period, region, or other element. 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.

An Access File is *required* to take advantage of intelligent partitioning. Intelligent partitioning places specific data values in each physical partition and uses 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 the ACCESS ddname name concatenation. On VM/ESA the Access File must have a file type of ACCESS. FOCSQL cannot be used.

FOCUS Access File Attributes

The Access File can include the following attributes:

Keyword	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 that identifies 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.

The syntax for an Access File is:

```
MASTERNAME filename DATANAME dataname1 WHERE test ;
[LOCATION locationname DATANAME dataname2]
```

where:

MASTERNAME

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

filename

Is the logical name of the Master File.

DATANAME

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

dataname1

Is the fully qualified physical file name of the data file, 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.

locationname

Is a one- to eight-character name consisting of letters, digits, and underscore characters that identifies the logical name for a segment stored in its own data file.

dataname2

Is the fully qualified physical file name of the data file in which the segment is stored, in the syntax native to your operating environment.

Two-Gigabyte FOCUS Database Implementation Guide

The following is a sample Access File:

```
MASTERNAME filename
- * The following is for the first horizontal partition.
DATANAME horizontal partition 1 name
[WHERE where test ;]
[LOCATION segment location name
- * The following is for the second horizontal partition.
DATANAME horizontal partition 2 name
[WHERE where test ;]
[LOCATION segment location name
```

Describing a Physical Database File

In the simplest case, a Master File contains no separately stored segments. If the corresponding data file has the same file name as the Master File with the default file type, extension or data set, no Access File is required for that database.

However, if the data file has a different name from the Master File, you need an Access File to identify the database file.

The Access File declaration is

```
MASTERNAME filename
DATANAME dataname
```

where:

filename

Is the logical name of the Master File.

dataname

Is the fully qualified physical file name of the data file, in the syntax native to your operating environment.

Describing a LOCATION File

Any segment in the database can be stored in its own separate data file. The Master File declaration for any separately stored segment must include a LOCATION attribute that assigns a logical name to the separate file:

```
FILENAME=filename, SUFFIX=FOC, ACCESSFILE=ACC1,$
SEGNAME=SEG1, LOCATION=locationname, $
```

The corresponding Access File, ACC1, associates this logical name with the name of the physical data file. The syntax in the Access File is

```
MASTERNAME filename
DATANAME dataname
LOCATION locationname DATANAME dataname
```

where:

filename

Is the logical name of the Master File, used in requests.

dataname

Is a fully qualified physical file name that contains the segment, in the syntax native to your operating environment.

locationname

Is the logical name of the separately stored segment, from the LOCATION attribute in the relevant Master File declaration.

dataname

Is a fully qualified physical file name that contains the segment, in the syntax native to your operating environment.

Describing a Horizontal Partition

A horizontal partition is a separate database file for specific field values in a database, such as year, date, or region. The Access File describes how to associate the Master File with its separate database files. It can also supply a predicate (WHERE criteria) for each partition, to identify the data values stored in that partition; FOCUS uses this predicate to optimize retrieval.

The syntax is

```
MASTERNAME filename
DATANAME dataname1 [WHERE test1];
.
.
.
DATANAME datanamen [WHERE testn];
```

where:

filename

Is the logical name of the Master File used in requests.

dataname1...datanamen

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

test1...testn

Are logical conditions, terminated with semicolons (;), that describe the data values in the corresponding partitions. Supported operators are EQ, GT, GE, LT, LE, FROM...TO, AND, and OR.

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

Two-Gigabyte FOCUS Database Implementation Guide

For example, the SALES database consists of partitions for the years 1992, 1993, and 1994:

```
MASTERNAME SALES
DATANAME fully qualified physical file name for sales 1994
DATANAME fully qualified physical file name for sales 1993
DATANAME fully qualified physical file name for sales 1992
```

The user references the three data files using the logical name SALES. For example:

```
TABLE FILE SALES
```

The next example provides a predicate that describes the values in each partition:

```
MASTERNAME SALES
DATANAME fully qualified physical file name for sales1994
WHERE SDATE FROM '19940101' TO '19941231';
DATANAME fully qualified physical file name for sales1993
WHERE SDATE FROM '19930101' TO '19931231';
DATANAME fully qualified physical file name for sales1992
WHERE SDATE FROM '19920101' TO '19921231';
```

With these predicates in the Access File, query performance increases dramatically because FOCUS examines the record selection criteria in each retrieval request and accesses only the specific partitions needed to satisfy the request. For example, in the following request, FOCUS does not access the physical file for sales 1992:

```
TABLE FILE SALES
PRINT *
WHERE SDATE GE '19930401';
END
```

Describing Locations in a Partitioned Database

In a horizontally partitioned database, each separately stored segment, or internally maintained index is also partitioned. The Master File and Access Files must list all the location files.

- In the Master File, the syntax is

```
FILENAME=filename, ACCESSFILE=accessname,$
SEGNAME=name1, ...[LOCATION=loc1],$
SEGNAME=namen, ...[LOCATION=locn],$
```

where:

filename

Is the logical name of the database.

accessname

Is the logical name of the Access File.

name1...namen

Are the names of the segments.

loc1...locn

Are the logical names of the files that contain the segment data.

- In the Access File, the corresponding syntax is

```

MASTERNAME filename
DATANAME maindat1
LOCATION loc1 DATANAME dloc11
.
.
.
LOCATION locn DATANAME dloc1n
DATANAME maindat2
LOCATION loc1 DATANAME dloc21
.
.
.
LOCATION locn DATANAME dloc2n
.
.
.
    
```

where:

filename

Is the logical name of the Master File.

maindat1

Is the fully qualified file name of the physical main data file for the first partition.

loc1...locn

Are the LOCATION names of the separately stored segments from the relevant Master File declarations.

dloc11...dloc1n

Are the fully qualified file names of the physical data files for all separately stored segments from the first partition, in the syntax native to your operating environment.

maindat2

Is the fully qualified file name of the physical main data file for the second partition.

dloc21...dloc2n

Are the fully qualified file names of the physical data files for all separately stored segments from the second partition, in the syntax native to your operating environment.

For example, consider the sales.mas Master File. The CUSTDATA segment is stored in a separate LOCATION file named MORECUST:

```

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

Two-Gigabyte FOCUS Database Implementation Guide

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

```
MASTERNAME SALES
DATANAME fully qualified physical file name for sales1994
LOCATION MORECUST
DATANAME fully qualified physical file name for more1994
DATANAME fully qualified physical file name for sales1993
LOCATION MORECUST
DATANAME fully qualified physical file name for more1993
```

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

Frequently Asked Questions

- ***What needs to be done to enable the new database?***

The FOCPARM profile must contain the following command:

```
SET FOC2GIGDB = ON
```

- ***Are there **MODIFY** issues to consider when implementing the 2-GB FOCUS Database?***

For a non-partitioned database in which the data file has the same file name as the Master File with the default file type, extension, or data set, there is nothing extra to do for MODIFY or Maintain to manipulate the database.

For a partitioned database, or a database where an Access File is specified, an explicit allocation command must be issued to link the Master File to the data file that will be manipulated. The explicit allocation command can be DYNAM, USE, or CLIST/JCL allocations on OS/390 or USE command in VM/ESA.

If the explicit allocation command is not issued, MODIFY and MAINTAIN will use FOCUS default behavior and create a temporary data set on disk.

- ***Are there **REBUILD** issues to consider when implementing the 2-GB FOCUS Database?***

The REBUILD command creates temporary files that may subsequently exceed the 2-GB FOCUS database limit. Therefore, it is recommended that you REBUILD in sections to avoid the need to allocate large amounts of space to ddname REBUILD.