bmc

Solution White Paper

Four Solutions to Manage Large IBM® IMS™ Databases

Implement best practices to dramatically increase IMS database size

Table of Contents

Executive Summary

As the volume of IBM® IMS™ data grows, the need for more efficient space management and data access increases. **If a database runs out of space, costly database downtime, recoveries, and unscheduled maintenance can have a significant impact.**

New IMS database development is scarce, but existing IMS databases continue to grow. If you have web-enabled legacy IMS databases, you have probably seen significant growth in them. Those databases will continue to grow, perhaps at a much faster rate than ever before.

This white paper discusses four ways to increase the size of IMS databases and considerations for each method.

WHAT TO DO WHEN YOUR IMS DATABASES GET TOO BIG

Traditional full-function databases are limited to 4 GB (VSAM) or 8 GB (OSAM). Once you reach those size limits, you have four options:

1. Convert from VSAM to OSAM

2. Compress IMS Data

3. Migrate Databases to Fast Path

4. Migrate Databases to Partitioning

The way you use the database can help you determine which path to take.

Note: Adding more data set groups (DSGs) and moving some segments to another DSG can provide short-term relief for space constraints, but this is not a recommended option because it will require more I/O.

1. Convert from VSAM to OSAM

The simplest way to double the size of your IMS databases is to convert the access method from VSAM to OSAM. VSAM databases are limited to 4 GB while OSAM databases can be up to 8 GB. Moving to OSAM may improve online application performance because of how IMS manages buffers for OSAM.

To complete the conversion:

1. Allocate OSAM data sets.
2. Unload the VSAM data set.
3. Implement a DBD change to indicate that the database now uses OSAM.
4. Reload to the OSAM data sets.

No application changes are necessary because the data structures and management by IMS remains the same.

This method works until databases approach the OSAM 8 GB limit. When your database is that large, you must compress data or migrate to a new database type (Fast Path or partitioned).

2. Compress IMS Data

Compression reduces cost by requiring less DASD for storing compressed data and reducing the load placed on I/O processing. Compression provides several benefits:

- Better buffer hit ratios for online transactions
- Smaller logs, which lead to smaller log volume, fewer log switches, fewer archive executions, and shorter change accumulation run times
- Dramatically lower costs for image copy space and storage
  - Recoveries can be completed faster because compressed image copies can be applied quickly.
- Improved utility maintenance time
  - For example, if it takes 30 minutes to reorganize a database with no compression, implementing a compression percentage of 50 percent can potentially shorten the reorganization to 15 minutes.
- Lower I/O costs
  - With compression, segments and data within buffers are smaller. Because the data being read into the buffer is compressed, you can allocate smaller and/or fewer buffers
- Ability to meet some security standards because compressed data cannot be displayed
- Improved online IMS application response time through efficient use of IMS buffers and virtual storage
- Reduced segment splits and related I/O
- Efficient free space utilization
- Reduced elapsed time and I/O activity for sequential batch IMS applications

Compression Techniques

Choose a compression technique that yields the best compression percentage for your data. Several vendor compression tools are available, and the various compression techniques (algorithms) yield different compression percentages. Test your data to

determine which compression technique works best for your environment. For example, the Huffman algorithm tends to provide the highest compression percentage for short, hierarchical data. On the other hand, the Ziv Limpel algorithm, which is used for hardware compression, works best with relational data and long segments.

Hardware vs. Software Compression

For IMS, accessing and using native hardware compression is not a viable option. IBM implemented hardware compression using a software compression product. Even if you choose hardware compression, you must implement software for IMS to communicate with the hardware.

Costs and Benefits

Compression provides many benefits, but it also has some costs, including the price of the compression software. IBM provides some basic compression software at no additional charge. Compressed data will lead to higher CPU costs because data must be expanded and compressed whenever an application needs it. Before implementing compression, balance the costs with the need for larger databases.

Compressed data must be able to be expanded. Vendors accept the responsibility for guaranteeing that compressed data can be expanded. Check with your vendors to see what data integrity checks they provide.

If compression does not resolve your database space issues, consider converting to Fast Path or a partitioned database.

3. Migrate Databases to Fast Path

If you have high application availability needs and compression doesn't solve your space problems, consider migrating to data entry databases (DEDBs). Fast Path is an included feature of the IMS subsystem, so every IMS installation has Fast Path available, although many IMS customers have never implemented it.

Fast Path processing, which includes the DEDB database type, was designed for high volume, high availability applications. DEDBs can have up to 2,048 4 GB areas (total of 64 TB), offer improved I/O, and can have secondary indexes. In DEDBs, all data is stored randomly, so the only way to store data sequentially is to implement a sequential randomizer.

Note: DEDBs do not support logical relationships or offline DL/I processing.

Fast Path supports high online transaction rates and delivers improved response times for database inquiries and updates. DEDBs can use expedited message handling that bypasses normal message queuing and scheduling.

Data Entry Databases (DEDBs)

DEDBs are similar to a hierarchical direct access method (HDAM) database and are stored in VSAM data sets, called areas. A DEDB always stores all of the segments that comprise a record in a single area (data set). Each area has an overflow area to accommodate frequent update activity. This design offers flexibility in storing, accessing, and most importantly, maintaining self-contained portions of a database.

Fast Path supports databases larger than 4 GB by enabling database partitioning. DEDBs can have up to 2,048 areas, and each area can have up to seven multiple area data sets (MADS).

DEDBs provide superior transaction throughput, manageability, large data volume capacity, speed, and lower overhead requirements than full-function databases. DEDBs process large volumes of data with a transaction rate higher than any other type of mainframe database. Fast Path DEDBs provide:

- MADS support to enable software duplexing of database areas
- Sequential dependents (SDEPs) for efficient journal-type applications
- Highly optimized I/O
- Virtual Storage Option (VSO), which enables a DEDB to be loaded into virtual main storage, thus improving performance because database segments do not have to be retrieved from DASD
- Two different locking mechanisms to reduce the amount of time an application waits for data
  - Normal mode is CI-level locking, which changes to UOW-level locking when an online utility is running.
- Shorter code path length for the application (up to 50 percent shorter)

- Reduced logging overhead during maintenance processes, providing the following benefits:
  - A “syncpoint complete” is issued when a database update is logged. This improves response time because a transaction verification message can be sent to a terminal before the update is written to DASD.
  - Database changes are held in the buffer pool until committed, reducing logging overhead and eliminating the need for batch change backouts.
  - Backouts are eliminated because only the “after image” of the database is logged.

4. Migrate Databases to Partitioning

Partitioning spreads database records across multiple partitions. Data set size limits apply to each partition, extending the capacity of the database as a whole. Partitioned databases enable:

- Parallel processing for routine database management tasks, such as reorganizations and batch tasks
- Reduced data retrieval times for partitioned indexes
- Reduced I/O contention for database resources

You can implement partitioning with **Partitioned Database Facility for IMS**, or you can migrate to a High Availability Large Database (HALDB). Both options require a reorganization with a DBD change.

The cost of converting to a partitioned environment depends on more than just the cost of the tool. Look for hidden costs in the time involved if you must redesign databases to achieve partitioning. Consider the IMS and system resources required for the partitioned database. For example, partitioning tools can create additional data sets that must be managed and maintained, and that can require more space than in a non-partitioned database. Additional log records can also be an issue.

Your requirements for secondary indexes, logical relationships, particular database types, and performance requirements may favor one solution over another.

Partitioning Options

Partitioned Database Facility for IMS provides a simple partitioning solution that can increase database size by a factor of 100. It fully supports secondary indexes (with no changes required), and it allows partitioned databases to maintain logical relationships to non-partitioned databases. Partitioned Database Facility for IMS supports HIDAM, HDAM, HISAM, SHISAM, and INDEX database types. It supports, but does not require, DBRC.

A HALDB enables large databases to have all of the features of full-function databases plus an almost unlimited size. Partitions also provide availability and maintenance benefits. For example, you can reorganize one partition while the remaining partitions remain online and available for processing.

The following chart compares the partitioning options:

Partitioned Database Facility for IMS	HALDB
DBRC is supported, but not required.	DBRC is required. MAXM Reorg products provide a RECON Substitution facility that enables you to use non-DBRC registered databases in a test environment.
Logical relationships are allowed to non-partitioned databases.	Logical relationships must be to other HALDBs.
Secondary indexes can be partitioned.	Secondary indexes must be converted to PSINDEX and must be loaded separately.
No application changes are required.	Application changes are required to take advantage of partition independence and for applications that read secondary indexes (8-byte/SX field).

Conversion Process

Converting to Partitioned Database Facility for IMS is straightforward, and the process is similar to adding data set groups. No changes are necessary to primary or secondary indexes. The included utilities can recommend key ranges.

Converting to HALDB requires that you determine the size and number of partitions and how to distribute the data. **MAXM Database Advisor for IMS** provides a wizard to help determine this information.

Determine the Size of the Partitions

HALDB allows up to 1,001 4 GB VSAM or 8 GB OSAM partitions. One of the main advantages of HALDB is that you can design partitions so you don't continually hit the size limit. If you want to keep each partition no larger than 2 GB, then set the initial size to 1.5 GB and allow for growth. You can choose different sizes for each partition.

Your SLAs are important when choosing partition sizes. For example, if your SLAs state you need to recover 1 GB of data in one hour, factor this into your partition size planning.

Determine the Number of Partitions You Need

It's important to design your HALDBs for maximum efficiency. Maintenance for HALDBs is managed at the partition level, meaning that you can take one partition offline for reorganization, backup, etc., while the other partitions are still available for application processing. However, there are times when you will need to process the entire database, for example, a reorganization to implement a structure change or a complete recovery.

Before you migrate to HALDB, consider the implications of managing an extremely large database and be prepared to manage the entire database at once. Do you have a batch window large enough to reorganize or recover a 10 TB database? Choose a manageable number of partitions so when you need to reorganize, back up, or recover the entire database, you can do so methodically and effectively.

Determine How to Distribute the Data

HALDB makes it easier to manage large databases because you can store data in the areas of the database that make the most sense for your environment. For maximum efficiency, carefully choose how to distribute your data. Consider what data is static and what data will change. The following examples show how you could choose to distribute data across partitions:

- A customer account database could be divided alphabetically (one partition for last names beginning with A, one partition for last names beginning with B). In this instance, it is likely that the data for partition Q will be much smaller than the data for partition S. Will having partitions of different sizes present issues for you?
- A billing database could be divided by time increments (all January activity in one partition, all February activity in one partition). For retail applications, the partitions for November and December may need to be larger than the partitions for other months.
- A database that is affected by different state laws could be divided by state (all Texas accounts in one partition, all Louisiana accounts in one partition).

Because you can distribute data as needed, you can also purge data easily. You can set up routines to purge all data in partition one when it meets your purge criteria (when partition one is more than twelve months old). You can save time and resources by purging data automatically.

Determine How HALDB Affects Your Test and Development Environments

HALDB introduced a new level of complexity in development/test environments. In a full-function world, developers can each have their own copy of a database to work with and those databases are refreshed by simply copying existing data. This is enabled by not defining test/development databases to DBRC. HALDBs must be registered to DBRC because part of the database definition resides in the RECON data sets. You must choose among the following:

- Providing developers with separate RECONs
- Having all developers work off a single database definition
- Using full-function equivalents that are not registered to DBRC (enabled by the RECON Substitution Facility provided with MAXM Reorg solutions)

Because the functionality and program return codes are nearly identical, most development/testing can be performed on a

full-function equivalent, and you can use a HALDB to confirm that all is correct in your system test environment before moving the code into production.

Database Structure and Planning Requirements

The types of supported databases, the number of data sets required, and supported database features can make a difference in the type of tool required to partition an IMS database. The following table shows the database types and features supported by Partitioned Database Facility for IMS and HALDB.

Support for	Partitioned Database Facility for IMS	HALDB
Maximum data set size	4 GB for VSAM and KSDS 8 GB for OSAM	4 GB for VSAM and 8 GB for OSAM (with restrictions)
Number of data sets per data set group	10	10
Maximum number of partitions and data set groups	127	1,001
Secondary indexes	✓	✓
Logical relationships	✓	✓
Virtual paired relationships	×	×
HDAM databases	✓	✓ (PHDAM)
HIDAM databases	✓	✓ (PHIDAM)
Root-only HISAM databases	✓	×
SHISAM databases	✓	×
Index build	✓	✓ (With MAXM Reorg products)
Index per partition	✓	✓
Index list data set (ILDS)	×	✓
Extended pointer	×	✓
Self-healing pointers	×	✓
Dynamic allocation	✓	✓
New IMS status codes	×	✓
DBRC required	×	✓ (MAXM Reorg products provide a RECON Substitution Facility that enables you to use non-DBRC registered databases in a test environment.)
Partition independence	×	✓
Image copy indexes	✓	✓

If you convert to a partitioned database (Partitioned Database Facility or HALDB), you must convert virtually paired logical children to physically paired relationships. MAXM Reorg products can generate an unload file that contains actual unload segments for VLCs (virtual logical children).

Index Support

Partitioned Database Facility for IMS supports both recoverable and non-recoverable primary and secondary indexes. You can restore the primary and secondary index using image copies and point-in-time recovery, or you can rebuild the index. The HALDB primary index (PSINDEX) and ILDS must be non-recoverable and cannot be image copied by IMS. If you have any procedures that use IMS image copies of primary indexes, you'll need to change those procedures when converting to HALDB. The ILDS must also be considered for HALDB image copy or point-in-time recovery processes.

No changes are required for secondary indexes when you convert to Partitioned Database Facility. The IDCAMS definition for a secondary index is the same as for non-partitioned databases. The size of the secondary index does not increase when converting the target database to Partitioned Database Facility for IMS.

HALDB requires that secondary indexes be converted to HALDB PSINDEXes when the database is converted to HALDB. The size of the PSINDEX record is increased by the 28-byte extended pointer set (EPS) and the size of the root key. The /SX field, which is part of the key, will also increase from 4 bytes to 8 bytes. The size of the HALDB secondary index increases dramatically for HALDB.

Logical Relationships

Both Partitioned Database Facility for IMS and HALDB support logical relationships and require that virtually paired logical children be converted to physically paired relationships. The logical parent concatenated key (LPCK) must be physically stored with the logical child.

Partitioned Database Facility for IMS uses symbolic pointing to locate the logical parent. Pointer resolution is not required after the logically related database is reorganized. HALDB uses the extended pointer set (EPS) and self-healing pointers to update the EPS after a logically related database is reorganized.

Partitioned Database Facility for IMS allows logical relationships to non-partitioned databases. HALDB requires that all logically related databases be in HALDBs.

Parallel Processing

Batch and BMP applications that read large databases can improve their performance by reading partitions in parallel. Both Partitioned Database Facility for IMS and HALDB allow a database PCB to read only a selected partition by adding a JCL statement to the batch or BMP application. Running an application for each partition in parallel greatly improves sequential read performance. The same can be done for batch or BMP programs that are driven by input files if the input can be separated for each partition to allow the programs to run in parallel.

Pointer Updates

HALDB self-healing pointers allow database or partition reorganization without updating the secondary indexes or logically related databases, which improves reorganization performance. However, the pointers in indexes and logical relationships must be updated and healed eventually, and that normally happens when the first BMP with update intent reads the database from the secondary index or logical relationship.

The pointer will not be healed until an application with update intent can update the pointer. If your applications read the secondary index only with read-only BMPs, you must run an update BMP to heal the secondary index. Databases with a high percentage of logical relationships between databases could experience performance problems with BMPs healing the pointers.

The number of locks dramatically increases when pointers are healed. The locks are not released until the application reaches a commit point. BMPs with update intent that do not take frequent checkpoints may hold so many locks that online performance can be affected.

Returning to a Non-Partitioned Format

Because to the increased size of the PSINDEX (up to three times the size of a non-partitioned secondary index) and the additional data sets and logging required by HALDB, returning to a non-partitioned database can be complex. Any application changes must also be backed out.

Partitioned Database Facility for IMS adds no data structures to the system, and backing out the partitions is as simple as reversing the conversion process. A DBDGEN and ACBGEN are required. Because no application changes are made to convert to Partitioned Database Facility for IMS, there is no need to consider applications in the return.

Partition Independence

HALDB enables partition independence while Partitioned Database Facility for IMS does not. With Partitioned Database Facility for IMS, DBRC authorization is done for the entire database. HALDB DBRC authorization is done for each partition, and you can DBR a partition to be used by database utilities while the other partitions are available online.

You might want to change your applications to take advantage of partition independence. DEDBs have always had area independence, and applications were designed to take advantage of the FH status code (area not available). Full-function applications were not designed for an equivalent "database record not available" status code.

An application does not know that a HALDB partition is unavailable until it makes a call to the partition and receives a BA status code. Most applications do not issue the INIT command for extended status codes, so the call to an unavailable partition will result in a U3303 pseudo-abend.

Archiving the Database

If you use database reorganization to archive and purge inactive segments, be aware that HALDB does not update secondary indexes during reload. Source segments cannot be deleted during reorganization because HALDB secondary indexes contain index records for source segments that have been deleted during reorganization, and deleting the source segments would result in an index pointer error. This design precludes the common practice in which segments are deleted from the database during reorganization by a user exit or by modifying the unload data set.

Another archive approach is to keep the most current data in the first (or last) partition and then age the data by incrementing partition numbers during reorganization, with the oldest partition data being discarded or archived.

Which Partitioning Option Is Right for Me?

Before moving to partitioning, consider the benefits (space and availability) versus the costs (maintenance of a very large database).

Partitioned Database Facility for IMS may the best option if you have:

- Logical relationships and don't need to migrate all related databases to partitioning at the same time
- SHISAM databases that cannot be changed

HALDB may be the best option if you need:

- Partition independence (the ability to perform maintenance on selected partitions while the other partitions remain online and available for processing)
- Extremely large databases

If You Convert...

If you decide to convert to a partitioned database, BMC can help. BMC solutions enable you to convert to a partitioned database in a single-step reorg. The solutions provide a wealth of functionality for managing all types of partitioned databases, including the ability to reorganize HALDB partitions in parallel. The BMC solutions allow the following modifications to all database types during reorganization:

- Change a data set block size
- Delete a segment from the end of a hierarchical structure
- Change a segment length or name
- Change a segment from fixed to variable length or from variable to fixed length
- Add or change a compression routine
- Change the randomizer program or name
- Change the number of root anchor points (RAPs) per block
- Change the number of bytes per record
- Change the size of the root addressable area (RAA)

- Add or delete a partition
- Change partition boundaries
- Convert between access methods (OSAM/VSAM)

CONCLUSION

When your IMS databases reach their size limit, you can change the database structure, implement compression, or both. Choose your option based on how you use the data and how much time you have to change it.

Start with the simplest methods. Change the access method for full-function databases from VSAM to OSAM, and implement compression. You can do both of these at the same time. Converting from VSAM to OSAM and implementing compression do not require application changes, but they require a reorganization with a DBD change.

If these don't work, consider moving to Fast Path or partitioned databases. If you have some DEDBs already, converting other databases to Fast Path may be a good idea. Converting to a DEDB may require a few application changes though. For example, Fast Path requires a DLI checkpoint call at the end of processing. It is also a good idea to add certain status code checking. If your application requires logical relationships or offline batch processing, Fast Path is not an option.

Converting to Partitioned Database Facility for IMS requires no application changes. Converting to HALDB technically requires no application changes, but adding certain status code checking is recommended.

If you migrate to partitioning or DEDB, consider adding compression at the same time. This will give you the processing benefits of compression and also provide you even better space utilization.

Work with your vendors to implement the strategy that works best for you.

FOR MORE INFORMATION

To learn more about solutions for managing for your IMS environment, visit **bmc.com/ims**

211420