The Mainstream
An article from the IBM @server zSeries software newsletter

Find the current version of The Mainstream here **ibm.com**/software/zseries/mainstream

The Value of IMS High Availability Large Database (HALDB)

from The Mainstream, Issue 9 - 2004
The IBM @server zSeries and S/390 software newsletter

High Availability Large Database (HALDB) extends the value of IMS™ data and applications. For over 35 years Information Management System (IMS) has provided unsurpassed availability and performance for critical enterprise applications. During this time, installations have invested heavily in IMS applications and databases. HALDB extends the value of these investments by providing increased capacity and extended availability.

HALDB enhancements have been made without requiring changes in existing applications. This compatibility allows enterprises to leverage their existing systems and expand their use with minimal costs.

As its name implies, HALDB provides both high availability and large database support.

Large Database Support

HALDB extends the size limitations of previously existing database types by partitioning them. The structure of each partition is similar to a non-HALDB database. Application programs are unaware of the partitions. They process these databases without being aware of the partitions. Table 1 shows non-HALDB database types and their corresponding HALDB database types.

Non HALDB database type	Corresponding HALDB database type
HDAM (Hierarchical Direct Access Method)	PHDAM (Partitioned HDAM)
HIDAM (Hierarchical Indexed Direct Access Method)	PHIDAM (Partitioned HIDAM)
Secondary Index	PSINDEX (Partitioned Secondary Index)

Table 1. Non-HALDB and HALDB Database Types

PHDAM, PHIDAM, and PSINDEX databases may have up to 1001 partitions. Each partition is similar to an HDAM, HIDAM, or secondary index database. PHDAM and PHIDAM partitions may have from 1 to 10 data sets. Each data set may be up to 4 gigabytes. This capacity exceeds all known database requirements. Previous to HALDB support, users of IMS databases addressed size limitations by various means. These included purging unneeded data, using data compression, and spreading data across multiple databases. HALDB provides a simpler solution to the size problem. The database may be converted to HALDB without application modifications.

As more records are added to a database, more partitions may be created. The size is practically unlimited. Figure 1 shows how a database may be extended by splitting a partition into two partitions.

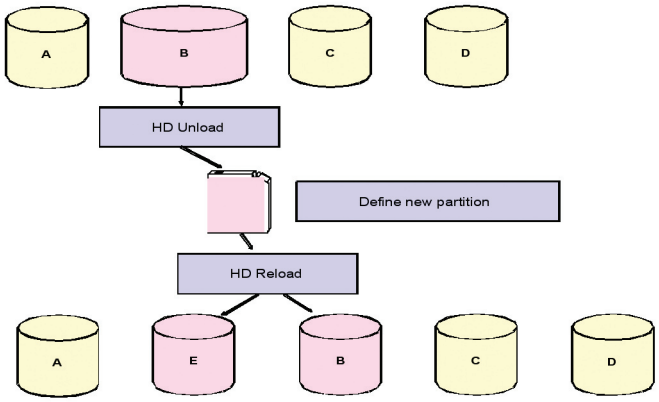

Figure 1. Adding a partition to a HALDB database

In Figure 1, partition B has grown. It is much larger than the other partitions. To provide more space in the database, the data in partition B is split between B and the new partition E. This is done by unloading partition B, defining the new partition, and reloading the data. The reloaded data is placed in partitions B and E. Partitions A, C, and D are unaffected by this process.

High Availability Support

As its name implies, HALDB enhances database availability as well as supporting larger database sizes.

Smaller Data Sets

Smaller database data sets provide availability and usability benefits. They may be copied faster than large ones. They may be recovered faster than large ones. These back up and recovery operations may be done in parallel. This reduces the elapsed time for these processes.

Reorganizations are shortened for the same reason. Partitions may be reorganized in parallel. Reorganizations gain another advantage from partitioning. Much of the update activity in some databases is concentrated in a small number of partitions. Reorganizations for all of the partitions of these databases are not required. Only those partitions with high update activity need to be reorganized frequently. This reduces the work of reorganization.

Self Healing Pointers

For many installations the most significant high availability benefit of HALDB comes from “self healing” pointers. Secondary indexes and logical relationships provide IMS with some of its unique application and performance benefits. These facilities use direct pointers to connect data in different records. These pointers provide fast and efficient data access. They provide great benefits, but they have costs. Secondary index and logical relationship pointers must be updated after a reorganization. When databases are reorganized, database segments are moved to new locations. The pointers must point to the new locations before they may be used. The reorganization of non-HALDB databases is followed by utility processes which produce corrected pointers. Logical relationship pointers are updated. Secondary indexes are rebuilt. This can be time consuming. For many databases, the updating of the pointers takes longer than the reorganization of the database. This is where self healing pointers provide an advantage to HALDB users.

Secondary index and logical relationship pointers are not updated during the reorganization of HALDB databases and pointers. This can significantly shorten the reorganization time. Of course, the pointers need to be corrected before they can be used. HALDB pointers are corrected when they are first used. The reorganization process creates records in a small data set, the Index List Data Set (ILDS). These ILDS records have the new location of the target segments. The pointer in the secondary index or logical related segment includes the last known location of the target and the reorganization number for the partition when the pointer was last updated. When using the pointer, IMS compares the reorganization number in the pointer and the reorganization of the target partition. If they are the same, the pointer is accurate and, therefore, it is used. If they differ, the pointer must be healed. The new location is found in the ILDS record and replaces the old pointer.

The following figures illustrate the self healing pointer process.

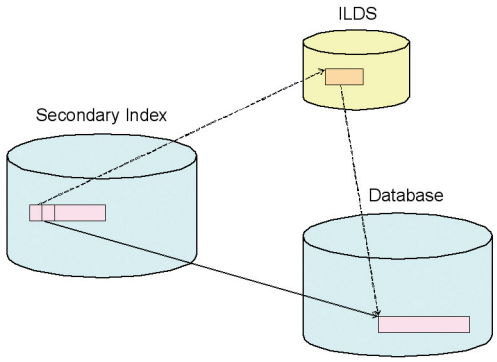

Figure 2. Database and secondary index before reorganization

Figure 2 shows the database before a reorganization. The secondary index entry has a pointer to the segment in the database.

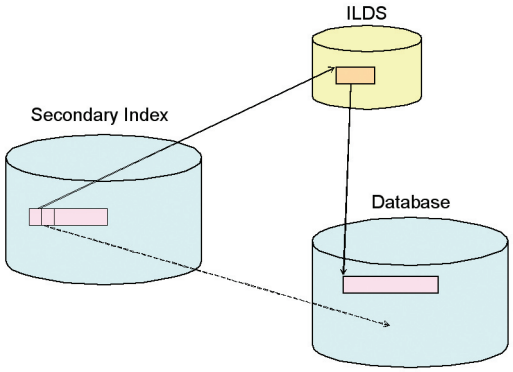

Figure 3. Database and secondary index after reorganization

Figure 3 shows the database after the reorganization, but before the pointer has been healed. The segment in the database has moved. The entry in the ILDS has been updated to point to the new location. The pointer in the secondary index points to the old location.

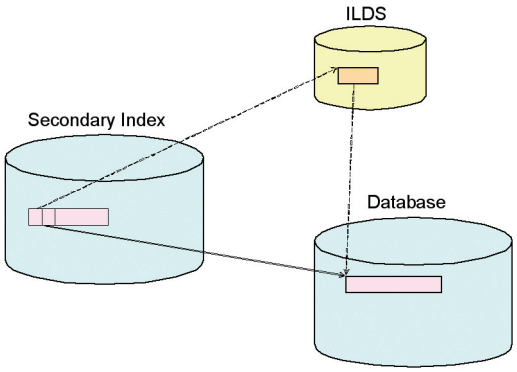

Figure 4. Database and secondary index after pointer healing

Figure 4 shows the database after the pointer has been healed. When an application used the secondary index entry, IMS compared the reorganization number in the entry prefix with that of the database. Since they did not match, IMS looked up the new location of the segment in the ILDS entry. It used this information to locate the target segment and to update the pointer in the secondary index entry. The secondary index entry now points to new location of the segment. Future uses of this secondary index entry will not have to access the ILDS.

There are multiple advantages to the self healing pointer process.

1. It eliminates much of the elapsed time for reorganization of databases with secondary indexes or logical relationships.
2. It does not waste resources updating pointers that are not used. Many secondary index entries and logical relationships are infrequently used. An example is a secondary index for a bank account which is based on the user's identification number, such as an American social security number. If the database is reorganized every month, a non-HALDB secondary index is rebuilt each month. If the database is HALDB, only the secondary index entries that are actually used during the month are updated.
3. The self healing process is efficient. The ILDS must be referenced to update a pointer. Its blocks (VSAM CIs) are read into database buffer pools. Each CI holds many entries. If there are a lot of pointer updates, these CIs are likely to remain in the buffer pool. This eliminates many potential reads of the ILDS.
4. If an installation wants to heal pointers before heavy processing of the database is done, users have this option. A BMP which references the pointers may be run. This will invoke the self healing process for these pointers. The BMP must have an update processing option (PROCOPT), but it does not need to update any records in the database.

HALDB Reorganizations

One of the primary benefits of HALDB is the shortening of reorganizations. The reorganization window is typically much shorter with HALDB. In fact, installations can reduce the window to almost any size by creating small partitions. Reorganization times are reduced for multiple reasons:

1. Partitions can be reorganized in parallel. Non-HALDB databases are reorganized by unloading the entire database and then reloading it. HALDB databases are reorganized by unloading and reloading the partitions. These may be done in parallel. Since each partition may be much smaller than the database, the reorganization can take much less time

2. When HALDBs are reorganized, their secondary indexes do not have to be rebuilt. For non-HALDB databases the time required for the rebuilding of secondary indexes can exceed that for reorganizing the indexed database. The database remains unavailable while the rebuilds are done. For HALDBs the self healing pointer process is used instead of rebuilding the secondary indexes.
3. When HALDBs with logical relationships are reorganized the logical relationship pointers do not have to be updated. For non-HALDB databases the time required to resolve logical relationships can exceed that for reorganizing the database. The database remains unavailable while the logical relationship pointers are resolved. For HALDBs the self healing pointer process is used instead of resolving the pointers during the reorganization outage.
4. Only partitions requiring reorganization are processed. Users have the option of reorganizing a subset of the partitions. If only some are disorganized, the time and resources to reorganize other partitions are not required.

IMS Version 9 Online Reorganization

IMS Version 9 introduces true online reorganization for HALDB databases. Partitions may be reorganized while they are being updated by IMS online systems and batch jobs. A partition remains available during all of the reorganization process.

Online reorganization takes advantage of the self healing pointer capability of HALDB. It reorganizes the data without updating every logical relationship or secondary index pointer during the reorganization. These pointers are updated when they are first used.

Online reorganization creates new database data sets. It reads database records from the existing data sets and writes them to new data sets. When all of the records in a partition have been copied, the old data sets may be deleted. While this copy process is being done, a cursor is used to track the location of records. If the cursor has passed a record, IMS knows it exists in a new data set. If not, it is in an old data set. This allows applications to access all of the data in

the partition while it is being reorganized. While records are being copied they are locked. When the copy is completed, the lock is released. This is the same type of locking that is done for applications. In fact, online reorganization behaves like a well designed application program. It locks a limited number of records at any time. It holds these locks for a limited time.

Since any full function database may be migrated to HALDB, HALDB online reorganization is the solution for 100% availability of data during reorganizations.

Application Compatibility

Application programs do not have to be modified when full function databases are migrated to HALDB. There are only two small exceptions. First, initial loads of databases with logical relationships cannot insert logical children. Logical children must be inserted as updates. Since very few installations initially load databases with logical relationships, this restriction is rarely encountered. Second, application programs which process secondary indexes as databases, not as indexes, may require modification. This is only required if the secondary index uses a /SX field to generate unique VSAM keys and duplicate data is included. The /SX field is expanded from four bytes to eight bytes in HALDB. This means that the offsets to the duplicate data fields are increased by four bytes. This exception is also rare. Most installations will have few, if any, application programs which will be affected. If they are, the required changes are simple. Only the offsets to the duplicate data fields need to be changed. Many installations require no application programming changes when migrating databases to HALDB. Others typically require a small number of minor modifications.

Application program compatibility allows users to take advantage of HALDB benefits with minimal costs.

HALDB Migration

A database may be migrated to HALDB by unloading it, redefining the database, and reloading it. The following steps are typically used:

1. *Plan the partitioning scheme:* Determine the number of partitions to be used and the keys which will be placed in each partition.
2. *Unload:* Unload the database using either the HD Unload utility or the High Performance Unload tool. Include control statements which specify that a migration to HALDB is being done. This creates unload records in HALDB format.
3. *DBDGEN:* Modify the DBD to specify a HALDB database type.
4. *Partition definition:* Define the partitions. HALDB provides an ISPF based utility for doing this. Alternatively, this may be done with DBRC commands.
5. *Data set allocation:* Allocate the database data sets.
6. *Partition initialization:* Initialize the partitions. HALDB provides a utility for this process. Partition initialization makes a partition usable. Once it is initialized, application programs may update it.
7. *Reload:* Reload the database using either the HD Reload utility or the High Performance Load tool. The input is the output of the unload in step 2.

Installations which have many databases to migrate can benefit from using the IMS HALDB Conversion and Maintenance Aid. It is an IBM® tool which helps automate the migration process. It may be used to analyze, model, and convert existing IMS databases to HALDB with minimal manual intervention.

Summary

As enterprises grow and the use of their data expands, databases grow. Performance and availability of these databases are critical due to the increased use of them by enterprises, their customers, and their suppliers. HALDB answers these needs. It supports database sizes beyond all known requirements while providing increased availability. It is the basis for future enhancements to IMS full function databases. HALDB online reorganization in IMS Version 9 is the first of these future enhancements.

Additional Information on HALDB

Redbook:

The Complete IMS HALDB Guide, All You Need to Know to Manage HALDBs, SG24-6945

http://www.ibm.com/redbooks

http://www.redbooks.ibm.com/redbooks/SG246945.html

http://www.redbooks.ibm.com/redbooks/pdfs/sg246945.pdf

Presentations:

IMS High Availability Large Database (HALDB)

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS185

Migrating to IMS HALDB

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS693

Application Design and Programming with HALDB

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS490

IMS HALDB Database Administration

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS842

Using GENJCL.USER to Allocate IMS HALDB Data Sets

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/TD100491

IMS Tools:

IMS HALDB Conversion and Maintenance Aid

http://www.ibm.com/software/data/db2imstools/imstools/imshaldb.html

© IBM Corporation 2004.

All rights reserved.

IBM, the IBM logo, IMS, are registered trademarks or trademarks of International Business Machines Corporation in the United States, other countries, or both.