IMS Version 7

High Availability Large Database (HALDB)

Rich Lewis
IBM Dallas Systems Center

This is an overview presentation of the High Availability Large Database (HALDB) capabilities introduced in IMS Version 7.

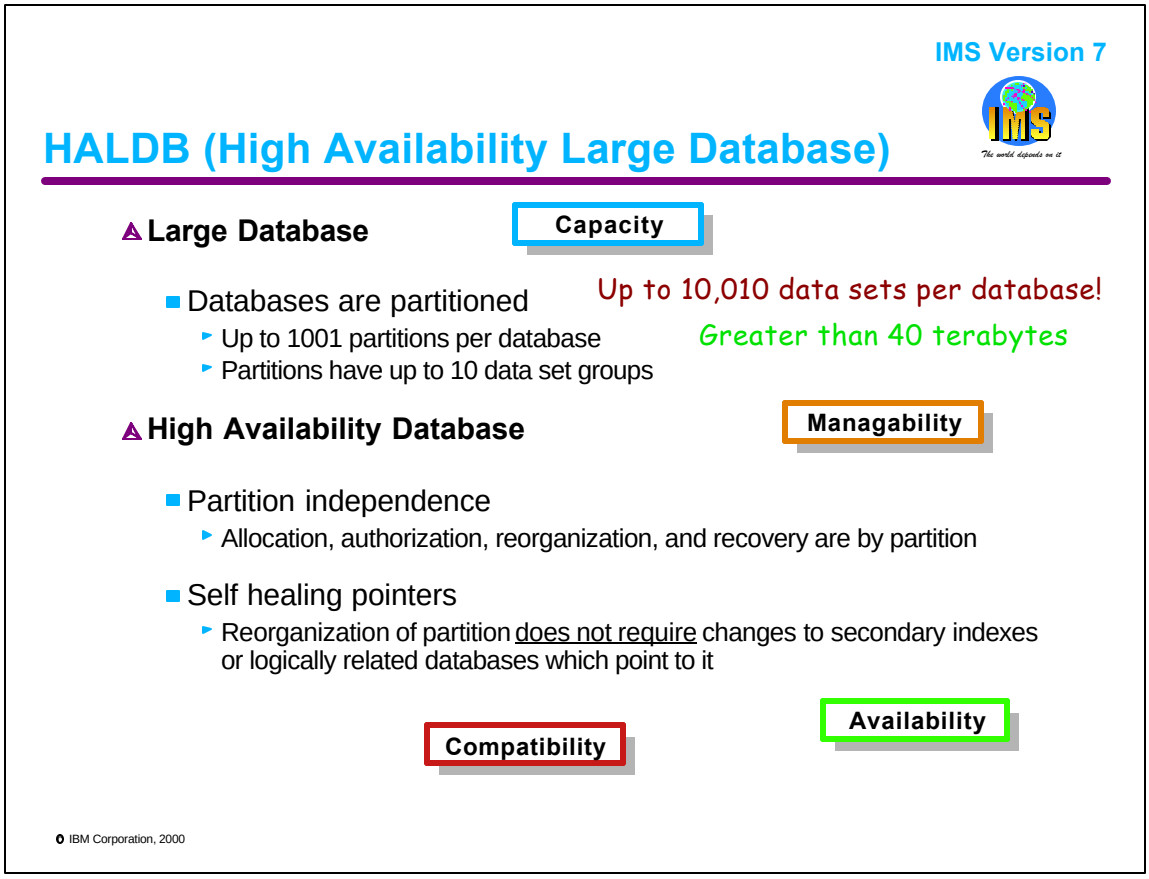
Abstract

IMS Version 7 introduces many enhancements to help its users provide increased availability, improved performance, and a more usable system.

High Availability Large Database (HALDB) is one of these enhancements. HALDB supports databases with up to 1001 independently managed partitions. The large number of partitions allows IMS full function databases to grow to over 40 terabytes.

HALDB provides greater availability through partition independence and parallel processing. Database processing, including maintenance activities, may be done in parallel. Each partition may be allocated, authorized, and reorganized independently. HALDB has a new pointer scheme for use with secondary indexes and logical relationships. This scheme eliminates some of the processing done by reorganizations.

This presentation is an overview of HALDB capabilities and benefits. Partitioning, pointer schemes, and migration considerations are included.



IMS Version 7 introduces a new capability for full function databases. This is High Availability Large Database (HALDB). HALDB databases have up to 1001 partitions. Each partition has up to 10 data set groups. This gives HALDB up to 10,010 data sets per database. Each of these data sets may be up to 4 gigabytes. So, the limit is 40 terabytes per database.

HALDB provides two availability benefits. First, partitions are managed independently. Each partition in a database may be allocated, authorized, reorganized, and recovered independently. Second, the reorganization of a partition does not require utilities to update the pointers in secondary indexes and logically related databases which point to the reorganized data. Even though the reorganization moves segments, pointers to those segments are not updated by the reorganization process. Instead, these pointers are updated as needed. This is a "self healing" process. This combination of capabilities can greatly reduce the windows required for database maintenance. Multiple partitions allow users to reorganize and image copy smaller amounts of data. This takes less time. The reorganizations may be done in parallel, as can the image copies. Since pointers are self healing, there is no need for utilities such as Prefix Resolution and Prefix Update, to correct pointers. This also reduces the time required for reorganizations.

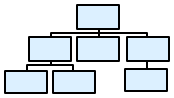
These new capabilities are delivered while maintaining application program compatibility. We will see how this is done.

The use of HALDB is optional. The non-HALDB databases of previous versions of IMS remain available with IMS V7.

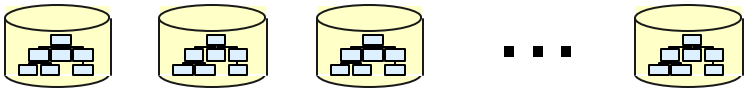
IMS Version 7

Highlights

- **New database types**
  - PHDAM - partitioned HDAM
  - PHIDAM - partitioned HIDAM
    - Index is also partitioned
  - PSINDEX - partitioned secondary index
- **Hierarchic structure is maintained**
  - A database record resides in one partition



- **Partition selection**
  - By key range or by user exit routine



HALDB introduces three new full function database types, partitioned HDAM (PHDAM), partitioned HIDAM (PHIDAM), and partitioned secondary index (PSINDEX). As the names imply, these are partitioned versions of the corresponding database types for non-HALDB databases. PHIDAM includes its primary index which is also partitioned.

HALDB databases have the same hierarchic structure that is used for other full function databases. A database record, which is a root segment and all of its dependents, resides in one partition.

Partitioning may be done either by key range or by a user written exit routine. Either method may be used with each of the three database types, PHDAM, PHIDAM, and PSINDEX.

IMS Version 7

Highlights

- **Logical relationships and secondary indexes are supported**
  - Secondary indexes may be partitioned
- **OSAM and VSAM (ESDS and KSDS) are used**
- **DBRC is required**
  - Databases must be registered
  - Dynamic allocation from DBRC information, not DFSMDA
- **Minimal (or no) application changes required**
  - Initial load cannot insert logical children (must be added by update)
    - New status code for load programs
  - 'Data unavailable' conditions apply to partitions
    - Database may be available, but partition unavailable

HALDB has complete full function database capabilities. This includes support for logical relationships and secondary indexes.

Like non-HALDB databases, HALDB databases use OSAM, VSAM ESDS, and VSAM KSDS data sets.

HALDB databases must be registered with DBRC. In fact, the definition process stores information about partitions in the RECONs. Since the information about HALDB data sets must be stored in the RECONs, this information is used for dynamic allocation. DFSMDA members cannot be used.

Most users will be able to convert non-HALDB databases to HALDB without any application program changes. Initial loads of HALDB databases cannot insert logical children. The logical children may be added by an update program which is run after the initial load. If an attempt to insert a logical child is made by initial load (PROCOPT=L), a status code of 'LF' is returned for the call. Data unavailable information in database PCBs and INIT DBQUERY calls, reports on the availability of the database, not its partitions. With HALDB a database may be available, but some partitions may not be available. That is, an installation may /DBR a partition while leaving the rest of the database available. Users who wish to take advantage of this new capability and who wish to have application programs react to it, may choose to modify existing programs. By including the the INIT STATUS GROUPx call or the EXEC DLI ACCEPT STATUSGROUP command in a program, the application will be returned a 'BA' status code when it attempts to access an unavailable partition.

IMS Version 7

Partition Independence

- **Commands**
  - Allowed on both databases and partitions
- **Availability**
  - Partitions are allocated and authorized independently
- **Scheduling**
  - Based on database availability
    - Partition may be unavailable with available database
- **Database Utilities**
  - Allowed on individual partitions or sets of them
  - Concurrent processing of multiple partitions allowed

HALDB partitions are managed independently. This is similar to the handling of areas with Fast Path Data Entry Databases (DEDBs).

Commands, such as /DBR DB, /START DB, and /DISPLAY DB may specify either a partition or a database. For example, a partition is unallocated when a /DBR DB is issued specifying a partition. If a database is specified, all partitions in the database are unallocated.

Partitions are allocated and authorized independently. For example, one partition may be authorized to a reorganization utility while other partitions in the database are authorized to an online system.

Scheduling is done on a database basis. Of course, IMS can schedule a program when a database in its PSB is not available. This remains true with HALDB. If a program examines a database PCB for availability information, issues an INQY DBQUERY call, an INIT DBQUERY call, or an EXEC DLI QUERY command, the availability of the database, not a partition is seen. If the database is available, but a partition is not available, the program will see the database's availability. Attempts to access the partition will receive a "data unavailable" condition. This is either a 'BA' status code or a U3303 abend depending upon whether the INIT STATUS GROUPx call or EXEC DLI ACCEPT STATUSGROUP command has been issued.

Database utilities may be run against the database, a partition, or a set of partitions. Many users will want to take advantage of the ability to execute against partitions to invoke parallel reorganization processing for the partitions in a database.

IMS Version 7

Definition Process

- **DBDGEN**
  - Used to define database
    - Segments, fields, hierarchic structure, data set group boundaries, pointer options, logical relationships, secondary indexes,...
- **HALDB Partition Definition Utility**
  - Used to define partitions in database
    - No. of partitions, partition selection, space characteristics, randomizers,...
  - ISPF based
  - Stores information in the RECONs
    - Definitions may be done with DBRC commands instead of this utility

HALDB databases and their partitions are defined in separate processes. The databases are defined with a DBDGEN. This is similar to non-HALDB databases. The DBDGEN includes the definition of the hierarchic structure, the assignment of segments to data set groups, the choice of pointer options, and the definition of logical relationships and secondary indexes.

The partitions for a HALDB database are not defined in the DBD. They are defined either with the HALDB Partition Definition utility or with DBRC commands. Partition information includes the number of partitions, the method of partition selection, the data set space characteristics, and the randomizer used with each partition. If partition selection is done by key ranges, the partition boundaries are defined here. If an exit routine is used, the routine is named here. Space characteristics include free space parameters used with each partition. The utility or commands store the partition definition information in the RECONs. There are new RECON records for holding this information.

IMS Version 7

Reorganizations

- **Reoganizations are simplified for logical relationships and secondary indexes**
  - **Work files** are **not used**
  - **Prefix Resolution, Scan, and Prefix Update** are **not used** to update logical relationship pointers
  - **HISAM Unload, HISAM Reload, or tools** are **not used** to update secondary index pointers
- **A new pointer scheme is used!**
  - Applies only to logical relationships and secondary indexes

The time required for reorganizations will typically be reduced when a database is converted to HALDB.

The utility processes for databases with logical relationships and secondary indexes are simplified. The use of "self-healing" pointers eliminates the need to run some utilities. Only the HD Reload and HD Unload utilities are required to reorganize a PHDAM or PHIDAM database. The Prefix Resolution, Scan, and Prefix Update utilities are never used with HALDB databases. When a PHDAM or PHIDAM database with secondary indexes is reorganized, the HISAM Unload, HISAM Reload, or tools replacements for these utilities are not used.

Each partition may be reorganized independently. Multiple partitions in the same database may be reorganized in parallel. One or more partitions may be reorganized while other partitions are not. For example, other partitions may remain allocated to online systems.

This combination of possibilities allows installations to meet various reorganization needs.

IMS Version 7

Direct and Indirect Pointers

- **HALDB uses both direct and indirect pointers**
  - Combination of pointers are used for logical relationships and secondary indexes
  - Direct pointers are RBAs of target segments
  - Indirect pointers "point" to Indirect List Entries (ILEs) in Indirect List Data Set (ILDS)
    - Indirect pointers are keys of ILEs (Indirect List Keys (ILKs))
    - An ILK is a token associated with the target segment
  - ILEs contain direct pointer to segment
    - Updated only by reorganizations
  - ILDS is a KSDS associated with a Partition

HALDB uses a combination of direct and indirect pointers for logical relationships and secondary indexes. The direct pointers are relative byte addresses (RBA) which point to segments. Indirect pointers "point" to Indirect List Entries (ILEs). These indirect pointers are actually keys of these ILEs. They are known as ILKs (Indirect List Keys). The ILEs are stored in an Indirect List Data Set. These ILDSs are new with HALDB. Each partition has an ILDS. The ILDS is updated only by reorganizations. It contains direct pointers to the new locations of the segments after the reorganization. The ILDS is a VSAM KSDS.

IMS Version 7

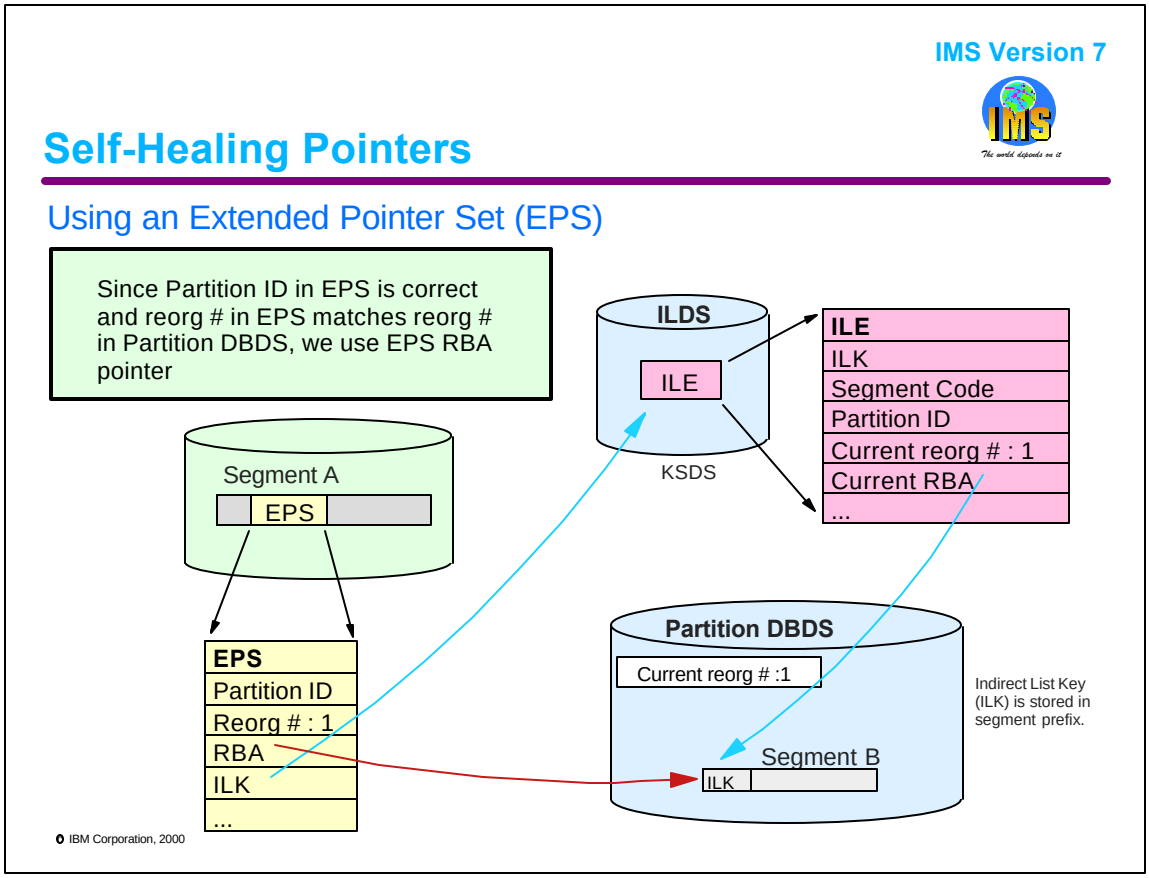
Extended Pointer Set

- **Extended Pointer Set (EPS) is used for logical relationships and secondary indexes**
  - Replaces direct or symbolic pointer used in Non-HALDB databases
  - Key of root is used to determine partition
  - EPS contains direct pointer, indirect pointer, reorganization number, and partition identification
    - If partition is correct and reorg number is current, direct pointer is used
    - If partition is wrong or reorg number is not current, indirect pointer is used
    - Indirect pointer points to Indirect List DS containing pointers from last reorg
  - EPS is not updated by reorganizations!
  - Direct pointer in EPS is updated when indirect pointer is used
- **Self healing pointers!**

An Extended Pointer Set (EPS) is used for logical relationship and secondary index pointers. It replaces the direct or symbolic pointer used with non-HALDB databases. The EPS has two "pointers". One is the direct pointer. The other is the indirect pointer. The EPS also contains information that is used to determine if the direct pointer is accurate. The direct pointer is the RBA of the target segment. It could be out of date. That is, it could point to the location of the target segment before the last reorg. The indirect pointer is the key of the Indirect List Entry (ILE) in for the target segment. The EPS also contains a reorganization number and partition ID. These were the partition ID and the partition's reorganization number when the direct pointer was up to date. If a reorganization has been done since this direct pointer was created or updated, the partition ID and/or the reorganization number will have changed. This information is used to determine which pointer to use.

When following a pointer, IMS first determines the partition in which the target segment currently resides. It does this by using the key of the target segment's root. This key is always stored in the segment with the EPS. IMS then compares the target partition's ID and reorg numbers with those stored in the EPS. If they match the direct pointer is used. If they both do not match, the indirect pointer is used to find the target segment's ILE in the ILDS.

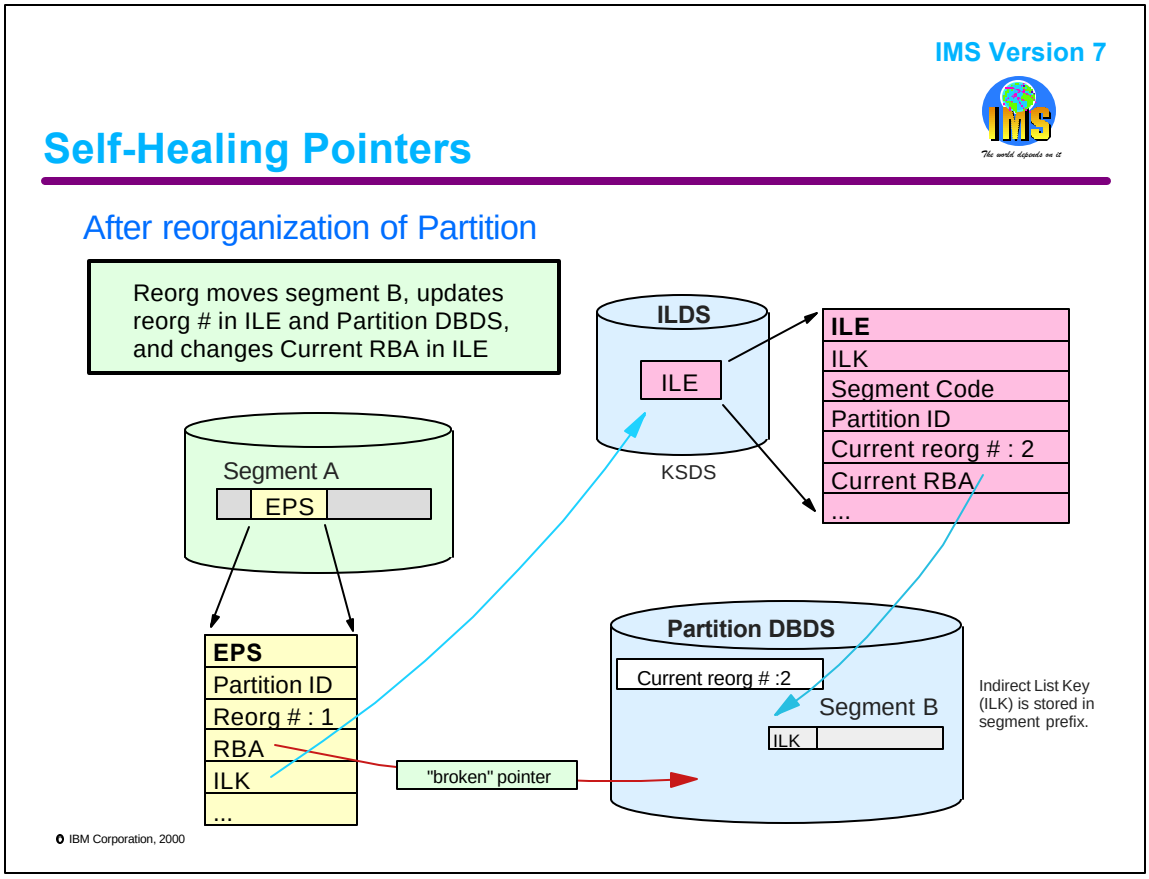
The EPS is not updated by reorganizations. Reorganizations update the ILEs in the ILDS. The direct pointer in the EPS is updated when the EPS is used. We will see how this "self healing" process works in the example which follows.



In this example segment A has a pointer to segment B. This pointer is either for a logical relationship or a secondary index. That is, segment A is either a logical child or a secondary index segment. Segment B is the target of the pointer. The Extended Pointer Set (EPS) is in the prefix of segment A.

The EPS contains a direct pointer to segment B. The direct pointer is a Relative Byte Address (RBA). The EPS also contains the partition ID of the partition where segment B resides. The EPS contains a reorganization number for the partition where segment B resides. In this case it is 1. The reorganization number is also stored in the partition database data set. When IMS opens the partition, it keeps the reorg number in its control blocks for the partition.

When the pointer from segment A to segment B is used, IMS first determines the partition where the segment currently resides. It does this by using the root key to find the partition. (With HALDB, segments with EPSs also contain the key of the target segment's root.) If this partition matches the partition ID stored in the EPS, the reorg number in the EPS is compared to the reorg number from the partition DBDS. If they are the same, the direct pointer (RBA) in the EPS is used to go directly to segment B and the ILDS is not used. In this case, the reorg numbers are both 1, so the direct pointer is used.



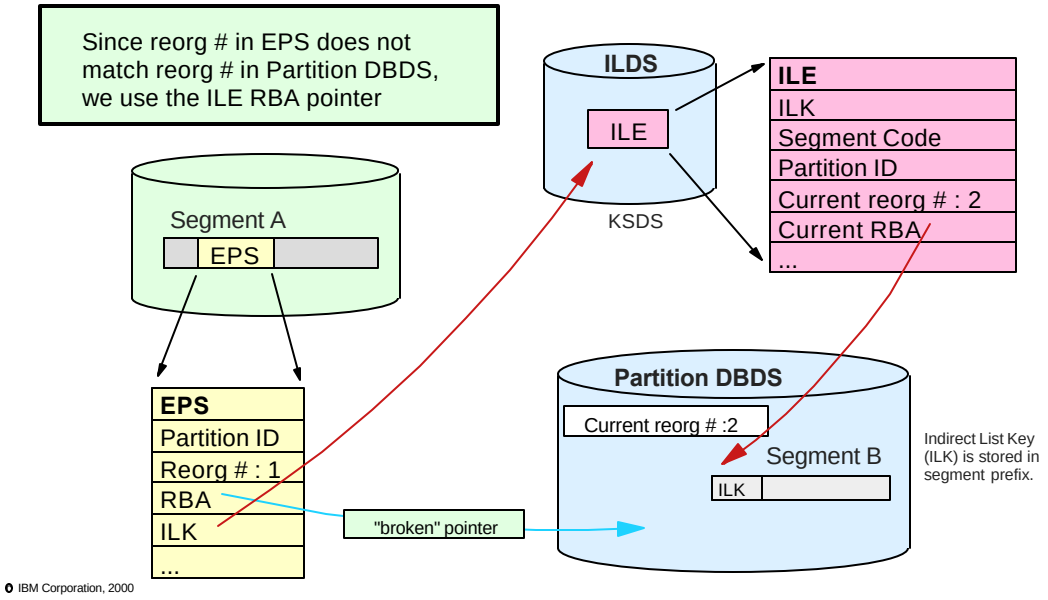
Reorganizing a partition changes the location of its segments. It also updates the ILEs for these segments in the ILDS and updates the reorg number stored in the partition DBDS. It does not update the pointers in EPSs. The ILE in the ILDS is easily found by the reload utility. The ILK, which is the key of the ILE, is stored in the prefix of the segment.

In this example, segment B is moved and its ILE is updated with an RBA which points to the new location of B. The reorg number in the partition DBDS is also updated.

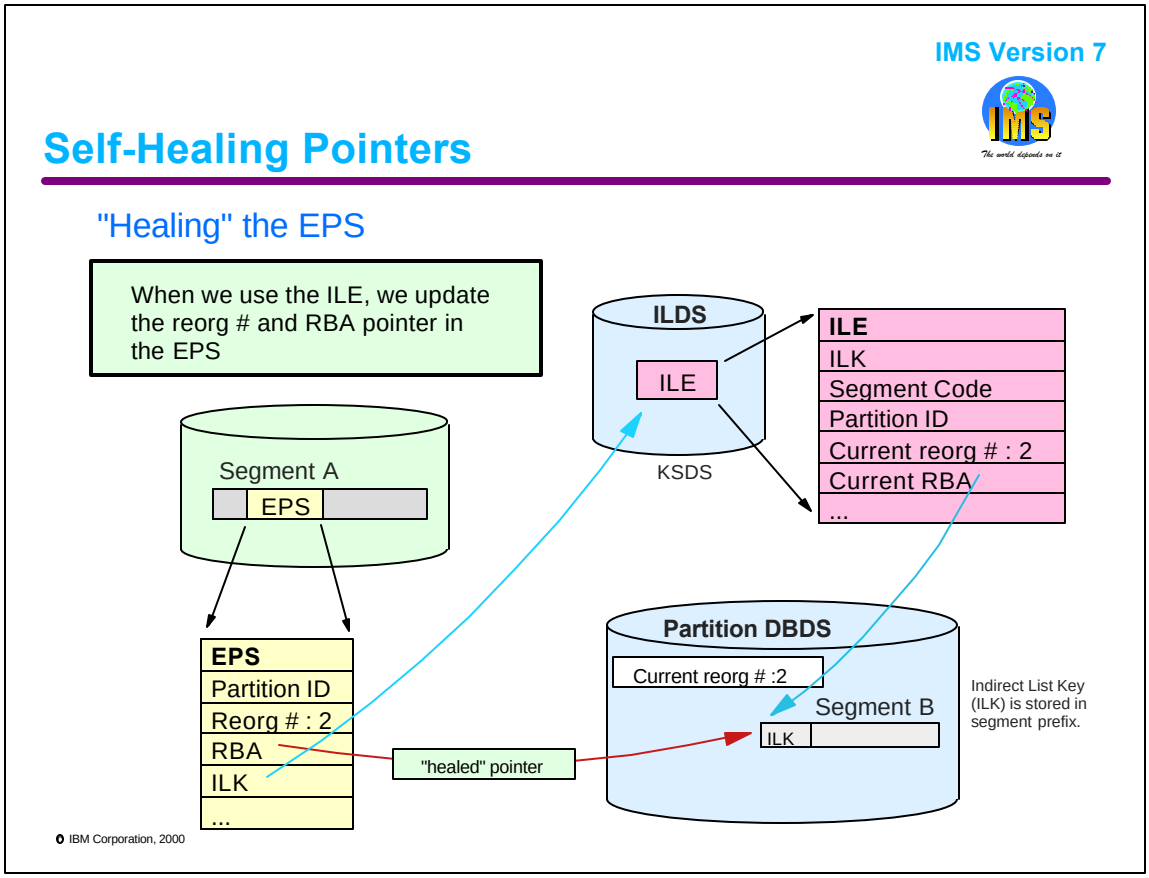
Since the EPS in segment A is not changed, its RBA does not point to the new location of segment B. It is now a "broken" pointer. The reorg number in the EPS remains 1.

Self-Healing Pointers

Using the EPS after the reorganization



An IMS call which needs to use the pointer from segment A to segment B will discover the "broken" pointer. IMS knows that the RBA pointer should not be used because the reorg number in the EPS does not match the reorg number in the partition DBDS. Instead of using the RBA, IMS will use the ILK in the EPS to find and read the ILE for segment B in the ILDS. The ILE contains a correct RBA pointer. It was updated by the reorganization.



When an ILE is used, IMS updates the direct pointer (RBA) and the reorg number in the EPS. This allows future uses of the EPS to avoid the overhead of referencing the ILDS. When this "healing" process completes, the EPS in segment A contains reorg number 2 and the RBA of segment B's new location. This update to the EPS is done only by programs which reference segment A with an update PROCOPT.

IMS Version 7

Reorganization Frequencies

- **Reorganization frequencies may be changed**
- **Increased free space may reduce reorganization frequencies**
  - HALDB may allow users to increase free space
  - Increased free space may reduce need to reorganize
- **Reorganization frequencies may be increased**
  - Reorg windows are reduced due to elimination of utility steps and parallel processing
  - Selected partitions may be reorganized independently

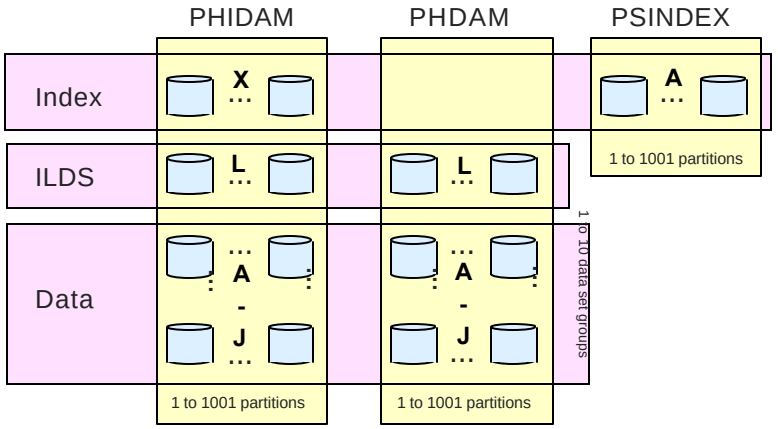
HALDB allows installations to reevaluate their reorganization frequencies.

Since HALDB databases may be spread over a much larger area, free space may be increased. Additional free space may reduce the requirement to reorganize since segments will more likely go in their home block. The downward trend in DASD costs and the greater need for database availability will make this use of HALDB attractive in many installations.

On the other hand, HALDB may allow for more frequent reorganizations. Since the time required for reorganizations is often reduced with HALDB, the window required for reorganizations is typically shorter. Since selected partitions may be deallocated from online systems, reorganizations of various partitions may be done at different times. This is another reason why partitions may be reorganized more frequently.

IMS Version 7

Database Data Sets



The data sets in a partition have generated data set names and DDNAMEs.
Letters are used to distinguish them.

- X - PHIDAM index
- L - ILDS
- A through J - Data data sets
- A - PSINDEX

This picture illustrates the data sets that may exist with HALDB databases.

PHIDAM, PHDAM, and PSINDEX (secondary index) databases may have from 1 to 1001 partitions.

PHIDAM databases have an index data set for each partition. They have an 'X' in their DDNAMEs and generated data set names. Similarly, PSINDEX databases have an index data set for each partition. They have an 'A' in their DDNAMEs and generated data set names.

PHIDAM and PHDAM partitions have an Indirect List Data Set (ILDS) for each partition. They have an 'L' in their DDNAMEs and generated data set names.

The database segments for PHIDAM and PHDAM databases are stored in data data sets. Each partition has its own data data sets. Each partition in a database will have the same number of data data sets. These data data sets will have letters 'A' through 'J' in their DDNAMEs and generated data set names.

IMS Version 7

DDNAMEs and Data Set Names

- **DDNAMEs**
  - Partition name is basis for DDNAME
    - Up to 7 characters
    - Assigned in HALDB Partition Definition Utility or by INIT.PART command
    - Letter is used as suffix
- **Data set names**
  - Begin with data set name prefix for the partition
    - Up to 37 characters
    - Assigned in HALDB Partition Definition Utility or by INIT.PART command
    - Letter and Partition ID are used as suffix

HALDB DDNAMEs are generated from the partition names. The partition names are 1 to 7 characters. They are specified in the HALDB Partition Definition Utility or by the DBRC INIT.PART command. The DDNAMEs are formed from the partition name and a one letter suffix. The suffix indicates the function of the data set in the partition.

HALDB data set names are generated from the partition data set name prefixes. These prefixes are specified in the HALDB Partition Definition Utility or by the DBRC INIT.PART command. A prefix may be up to 37 characters. The prefix is combined with the letter associated with the data set and the partition ID. The partition ID is a five digit decimal number. Since the partition ID is different for each partition and is included in the data set name, each partition in a database could use the same data set name prefix. This could simplify the naming process. On the other hand, partitions could have unique data set name prefixes.

IMS Version 7

DDNAMEs and Data Set Names

Example: PHIDAM with 3 data set groups, FRANCE partition

Partition_name of FRANCE

DSN_prefix of IMP0.DB.INV23.FR

PartitionID of 00004

Data set	DDNAME	Data Set Name
Data set group 1	FRANCEA	IMP0.DB.INV23.FR.A00004
Data set group 2	FRANCEB	IMP0.DB.INV23.FR.B00004
Data set group 3	FRANCEC	IMP0.DB.INV23.FR.C00004
ILDS	FRANCEL	IMP0.DB.INV23.FR.L00004
PHIDAM Index	FRANCEX	IMP0.DB.INV23.FR.X00004

This is an example of the DDNAMEs and data set names used for a partition. This is an inventory database. The partition name is 'FRANCE'. The user has decided to include an identification of the database in the data set name. This is 'INV23'. The user has also decided to include an identification of the partition, but not the partition name, in the data set name. 'FR' is used for this identification. The data set name prefix for the partition is 'IMP0.DB.INV23.FR'. IMS has assigned a partition ID of 00004 to the partition.

IMS Version 7

HALDB Migration

- **Migration**
  - Uses Prereorg, HD Unload, and HD Reload utilities with new control statements
    - Prereorg has new function: initialization of data sets
  - Databases logically related to each other must be migrated together
    - Logical relationships between HALDB and non-HALDB databases are not allowed
  - Secondary indexes must be migrated with the databases to which they point
    - Only HALDB secondary indexes may be used with HALDB databases

IMS provides utilities to migrate full function databases to HALDB. New capabilities in the Prereorganization, HD Unload, and HD Reload utilities are used.

HALDB databases cannot be logically related to non-HALDB databases. This means that logically related databases must be migrated together.

Similarly, HALDB databases cannot have non-HALDB secondary indexes pointing to them and HALDB secondary indexes cannot point to non-HALDB databases. This means that all of the secondary indexes pointing to a database must be migrated to HALDB secondary indexes when the target database is migrated.

An IMS transaction or program may access a mixture of HALDB and non-HALDB databases. Only those databases which have relationships to each other must be migrated simultaneously.

IMS Version 7

HALDB Fallback

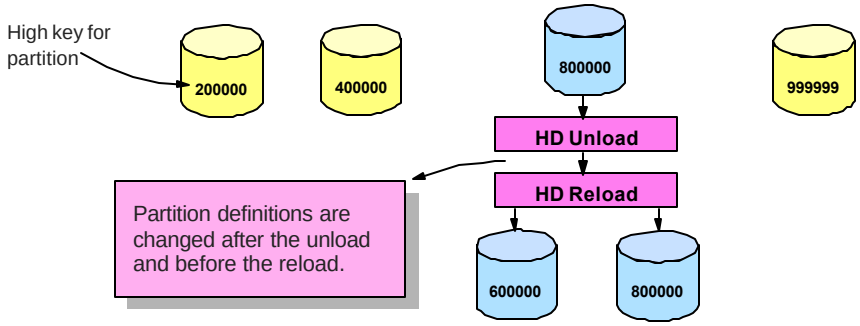
- **Fallback**
  - Fallback from HALDB to HIDAM, HDAM, and secondary indexes is supported
  - Uses Prereorg, HD Unload, HD Reload, Prefix Resolution, and Prefix Update utilities with new control statements

IMS provides utilities to fall back from HALDB to non-HALDB for one or more databases. This involves the use of Prereorganization, HD Unload, HD Reload, and the Prefix Update utilities.

IMS Version 7

Adding, Deleting, and Changing Partitions

- **Partitions changes**
  - Partitions may be added and deleted
  - Partition boundaries may be changed
- **Partition changes are made with HD Unload and HD Reload**
  - Only changed partitions are unloaded and reloaded
  - **Other partitions remain available during the process**



HALDB is designed to allow users to easily add partitions, delete partitions, and change partition boundaries. This is done by unloading and reloading the affected partitions. Affected partitions are those from which or to which database records are moved.

In the first step, the affected partitions are unloaded with the standard HD Unload utility. The change in partition definitions is done in the second step. Finally, the HD Reload utility is run in the second step. It loads the new, and/or changed partitions.

In the example shown here, the partition with high key 800000 is split into two partitions. The new partitions have high keys of 600000 and 800000. HD Unload is run against the original partition with high key 800000. The HALDB Partition Definition utility is executed to add the new partition with high key 600000. Finally, the HD Reload utility is run. Its input is the output of the HD Unload job. The other partitions may remain available for processing while these changes are made.

IMS Version 7

HALDB Support

- **HALDB is supported with:**
  - Data sharing
  - Remote Site Recovery (RSR)
  - Extended Recovery Facility (XRF)
  - Online Change
  - OSAM Sequential Buffering
  - IMS Monitor and IMS Performance Analyzer
  - …

HALDB has the same support that non-HALDB full function databases have. That is, they may participate in data sharing, RSR, XRF, and online change. HALDB database data sets may use OSAM sequential buffering and VSAM Hiperspace buffers. HALDB activity is monitored by the IMS Monitor and reported by the IMS Monitor and the IMS Performance Analyzer product.

IMS Version 7

HALDB

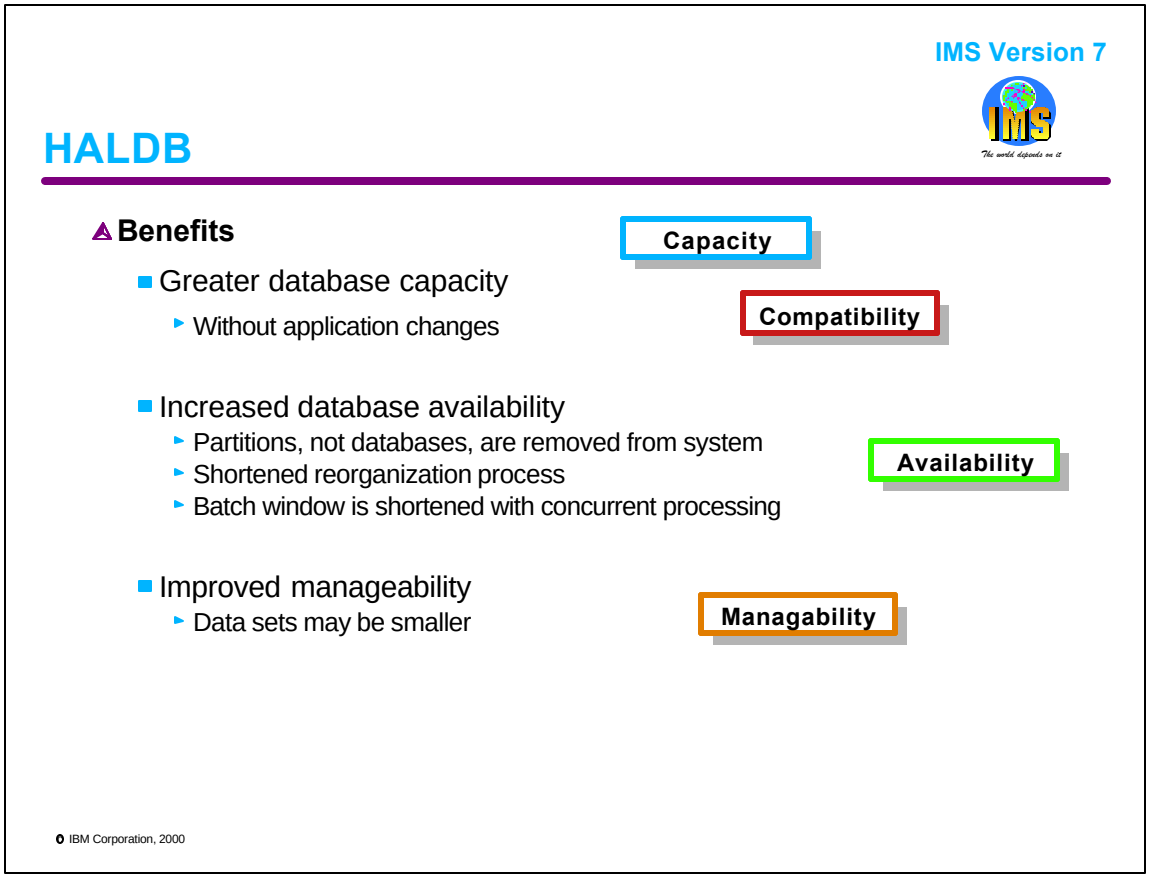
Database candidates for HALDB

- Very large databases
  - Approaching 4G (VSAM) or 8G (OSAM) limitations
  - Theoretical limit is now over 40 terabytes
- Medium and large databases
  - Parallel processing to meet time deadlines
  - Less frequent reorganizations due to more free space
- Any size database
  - More frequent reorganizations
  - Making only parts of the data unavailable for database maintenance

The most obvious candidates for HALDB are very large databases. Databases which are approaching the data set size limits of 4GB for VSAM or 8GB for OSAM have their size restrictions removed when they are converted to HALDB.

Medium and large sized databases with time deadlines for batch executions may benefit from the use of parallel processing against multiple partitions. Since space is practically unlimited with HALDB, more free space may be defined for databases. This may eliminate the need for some reorganizations or reduce their required frequencies.

Even relatively small databases are candidates for HALDB. Since reorganizations may require less time, they may be done more frequently. Installations that can benefit from removing only parts of a database for maintenance processing, will take advantage of HALDB.



HALDB benefits users in several ways.

HALDB allows databases to become larger without requiring changes to application programs.

Availability is increased by multiple means. HALDB allows partitions, not entire databases to be removed from systems for database maintenance purposes. Reorganization times are shortened by providing for concurrent reorganizations of partitions and eliminating utility step requirements to update logical relationship and secondary index pointers. Since partitions may be processed in parallel, HALDB may reduce batch and utility window time requirements.

Finally, HALDB eliminates the need to have very large data sets. This improves their manageability.