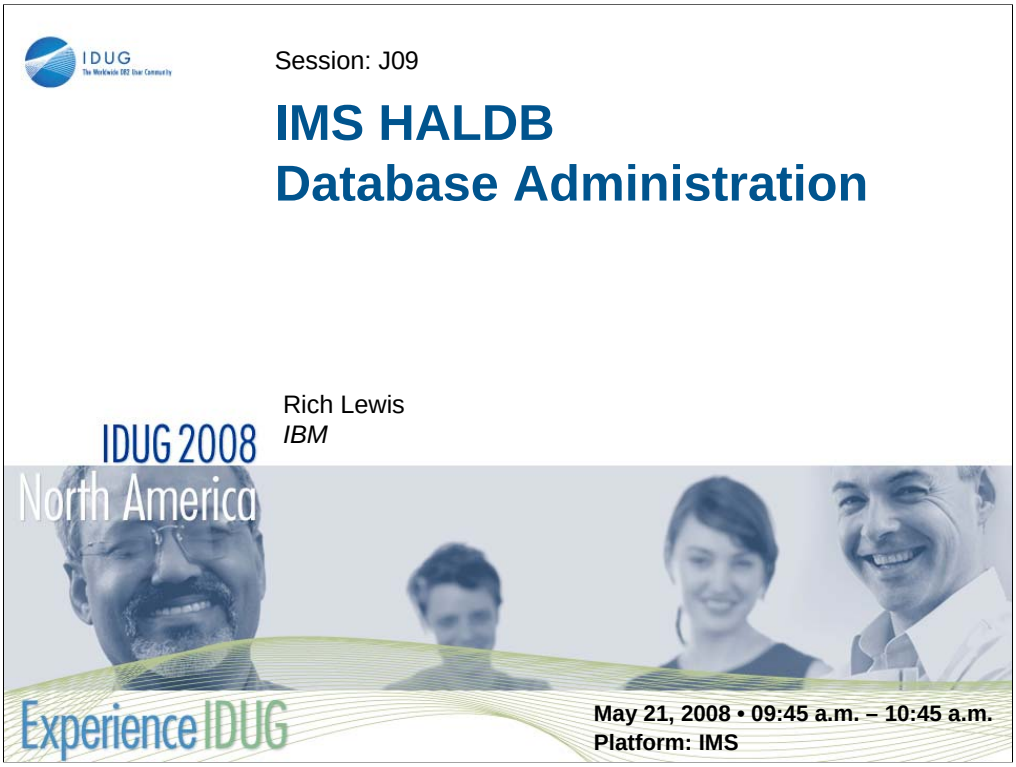
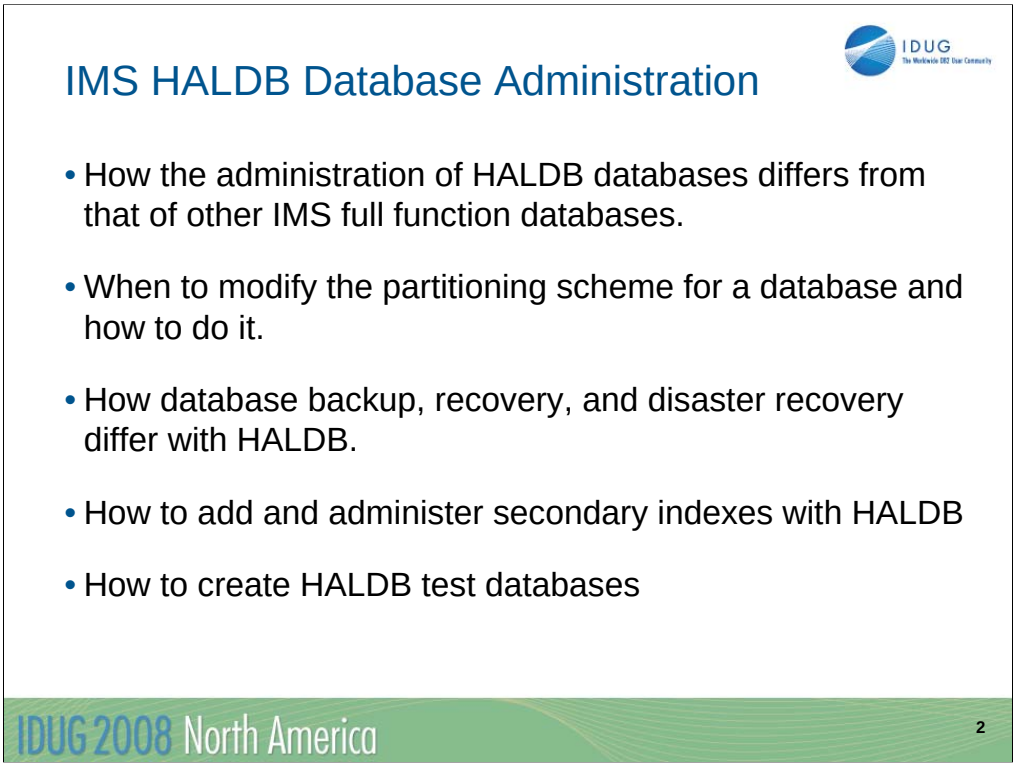


IMS High Availability Large Database (HALDB) provides increased flexibility with IMS databases. Database administrators have new options and new tasks with HALDB. This session describes these options and tasks. It includes partition initialization, partition sizing, partition modifications, backup and recovery, creating test databases, reorganization alternatives, secondary indexes, and performance options. Special attention will be paid to avoiding the problems that others have encountered. If you are responsible for administering HALDB databases, you will want to attend this session.

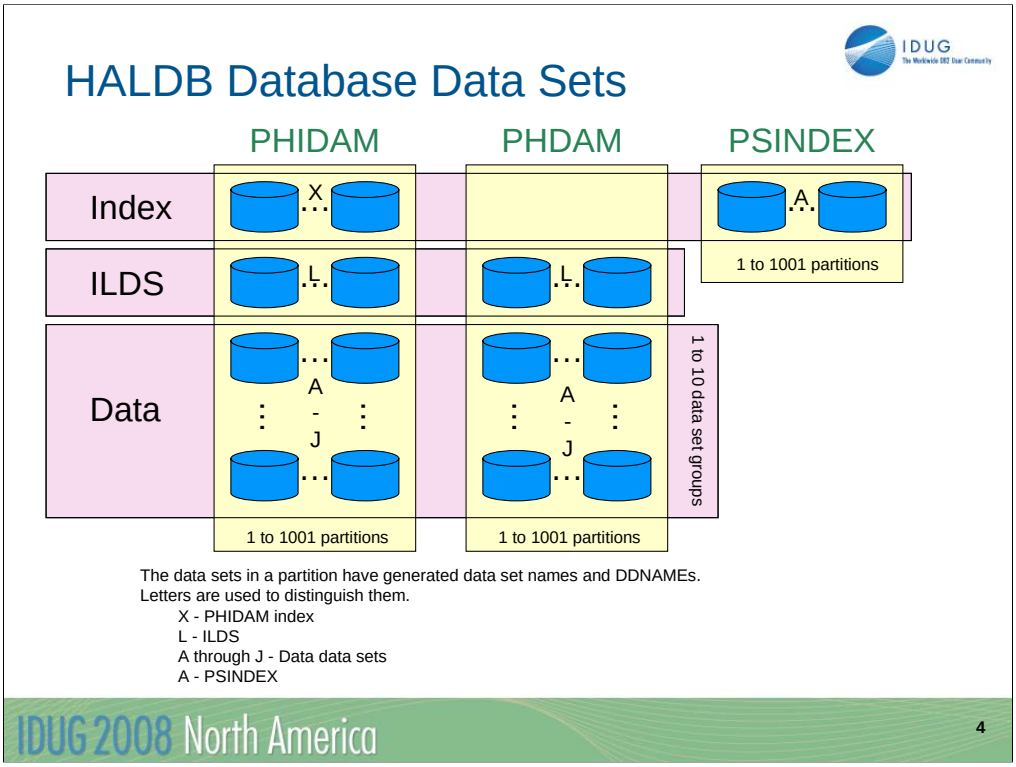


Agenda

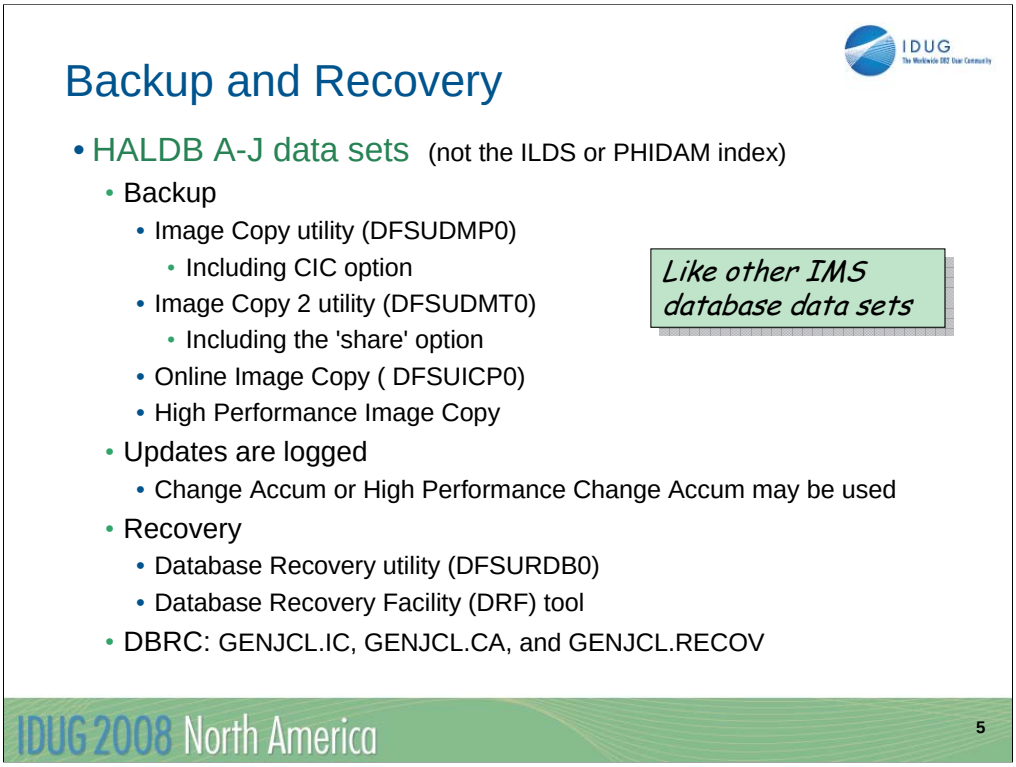
- Overview of HALDB data sets
- Backup and Recovery
  - Disaster Recovery
- Healing Pointers After Reorganizations
- Adding, Deleting, and Modifying Partitions
- Test Databases
- Secondary Indexes
- Performance
- Sources of more information about HALDB

IMS HALDB Database Administration

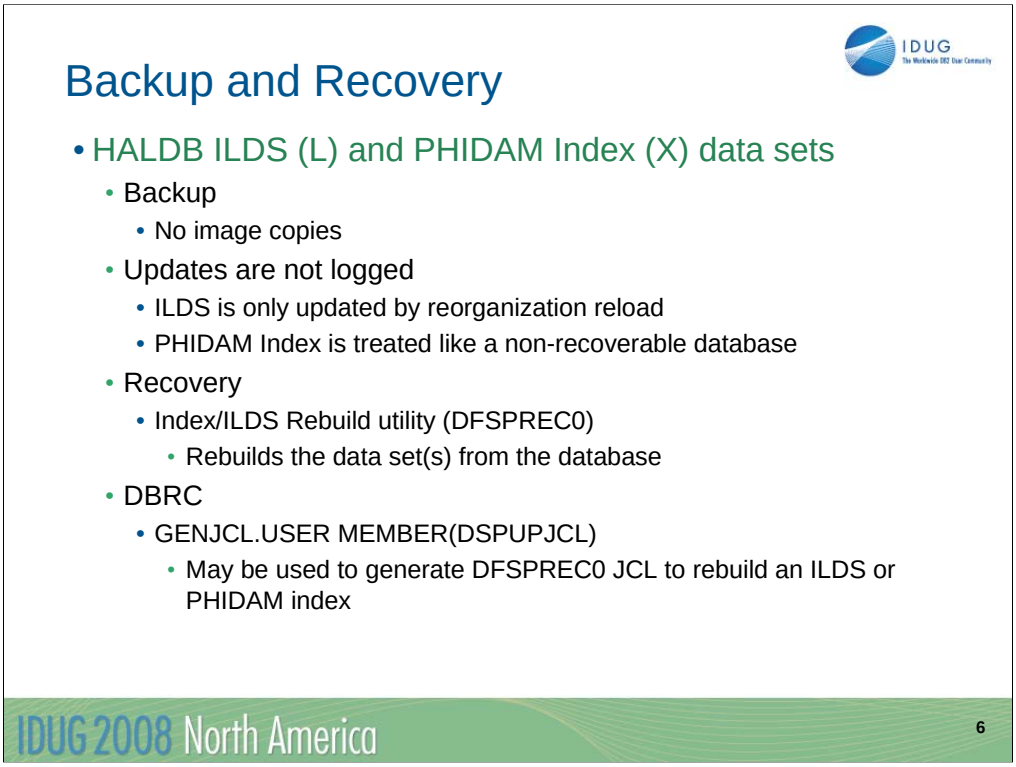
- Overview of HALDB data sets
- Backup and Recovery
  - Disaster Recovery
- Healing Pointers After Reorganizations
- Adding, Deleting, and Modifying Partitions
- Test Databases
- Secondary Indexes
- Performance
- Sources of more information about HALDB



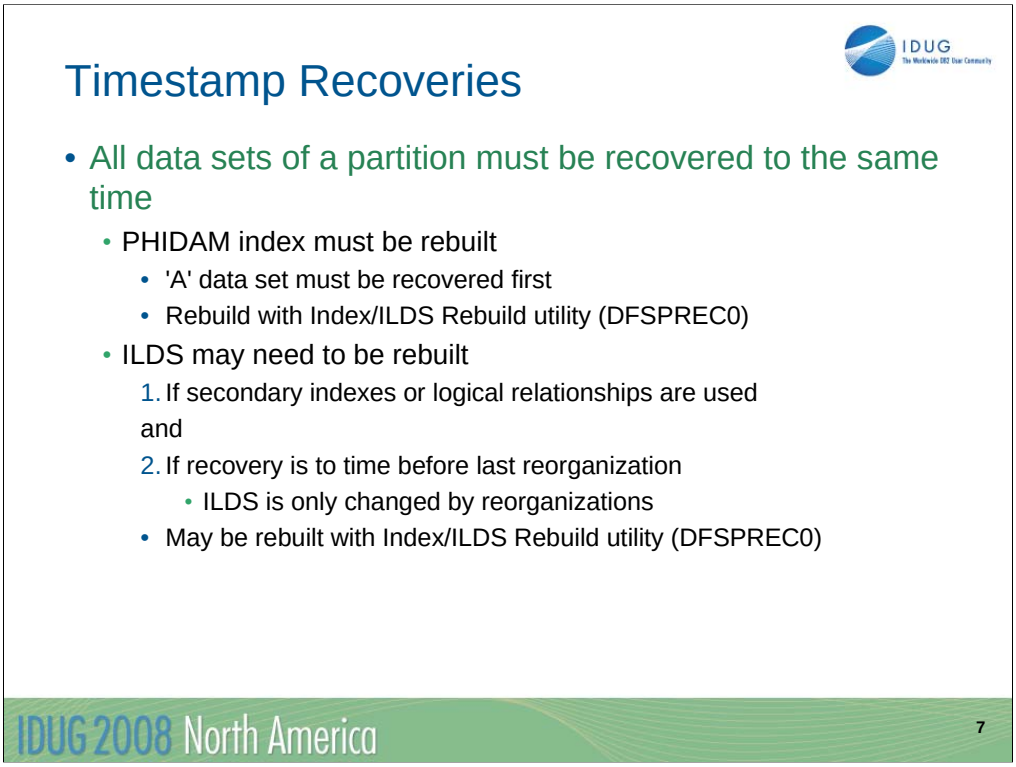
This is a review of the database data sets used by HALDB. This page shows a pictorial representation of the data sets used by HALDB. As indicated in the visual, each data set type has a letter associated with it. These letters are used in the data set names and DDNAMEs for these data sets.



The backup and recovery of most HALDB data sets is like that for non-HALDB database data sets. They are image copied with one of the image copy utilities. Updates are logged. These updates may be accumulated with Change Accumulation. The data sets are recovered with the Database Recovery utility, or the Database Recovery Facility tool. JCL for image copies, Change Accumulation, and database recovery may be generated with DBRC GENJCL commands. ILDS and PHIDAM index data sets do not use these processes.

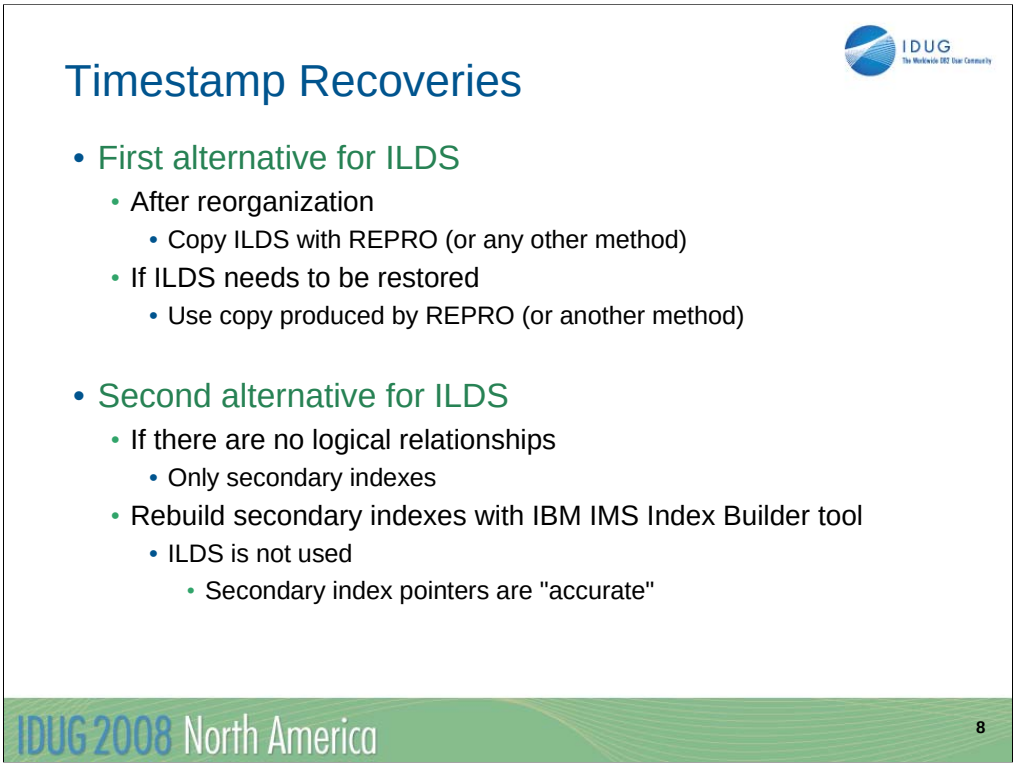


ILDS and PHIDAM index data sets are not image copied. Updates to these data sets are not logged. The ILDS is only updated by reorganizations. The PHIDAM Index is updated by inserts and deletes of root segments, but the changes to the index are not logged. The index is similar to a non-recoverable non-HALDB database. If the ILDS or PHIDAM index is lost, it is rebuilt using the Index/ILDS Rebuild utility (DFSPREC0). This utility rebuilds these data sets by reading other data sets in the partition. IMS supplies a skeletal JCL member which may be used with a GENJCL.USER MEMBER command to create the JCL for the Index/ILDS Rebuild utility.



Since PHIDAM indexes and ILDSs have special recovery processes, they also have special considerations for timestamp recoveries. When a timestamp recovery is done for a partition, its PHIDAM index must be rebuilt using DFSPREC0. The ILDS may also be rebuilt using DFSPREC0. Of course, if there are no secondary indexes or logical relationships, the ILDS is always empty and does not need to be rebuilt.

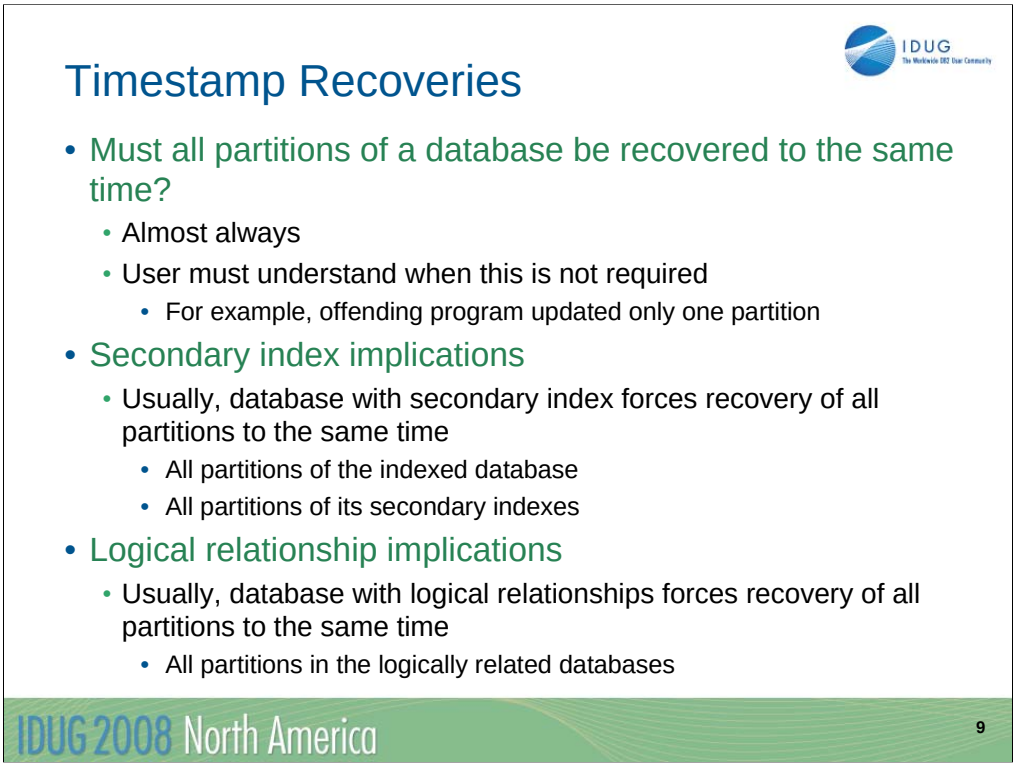
Since the only changes to the ILDS are done by reorganizations, the ILDS does not have to be rebuilt if the recovery is not to a time before the last reorganization of the partition



There are alternatives for the ILDS.

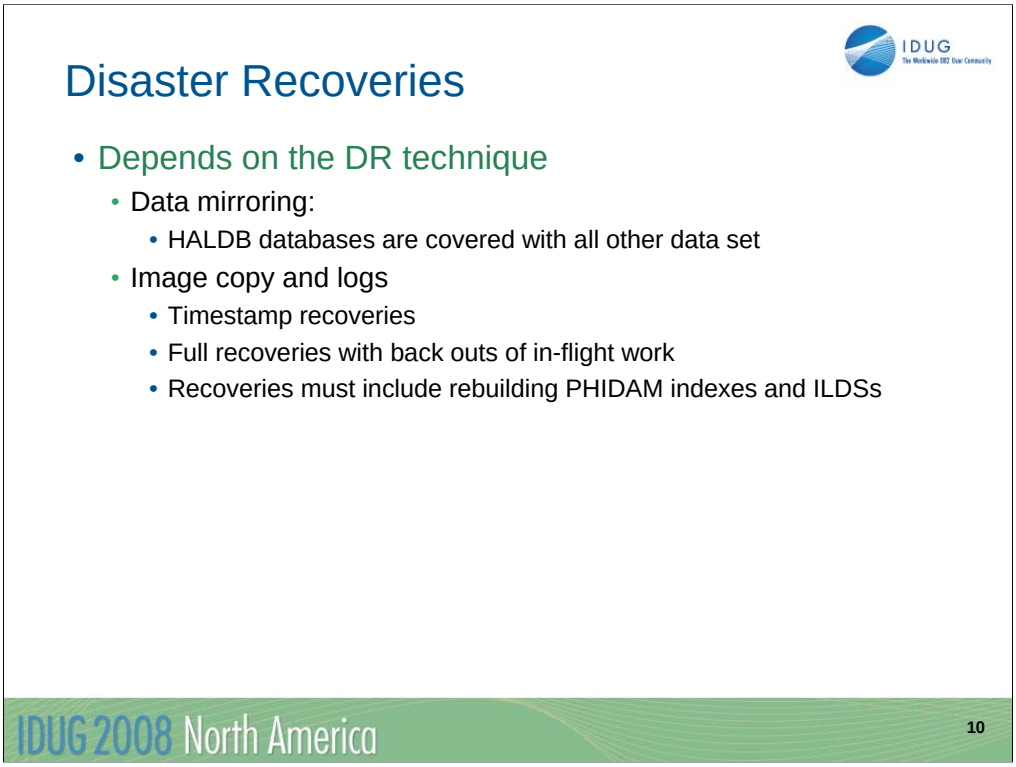
If the recovery is to a time before the last reorganization and you have a copy of the ILDS at that time, you may restore it. In this case, you do not need to rebuild the ILDS.

If you have a tool such as IBM IMS Index Builder, you may use it to rebuild the secondary indexes. If you do this, the pointers in the secondary indexes will be accurate and the ILDSs will not be needed for pointer healing.



HALDB allows you to do a timestamp recovery for a partition without recovering the other partitions in the same database. Even though it is allowed, it is rarely the correct thing to do. Usually, the entire database must be recovered to the same time. It is the responsibility of the user to understand this requirement.

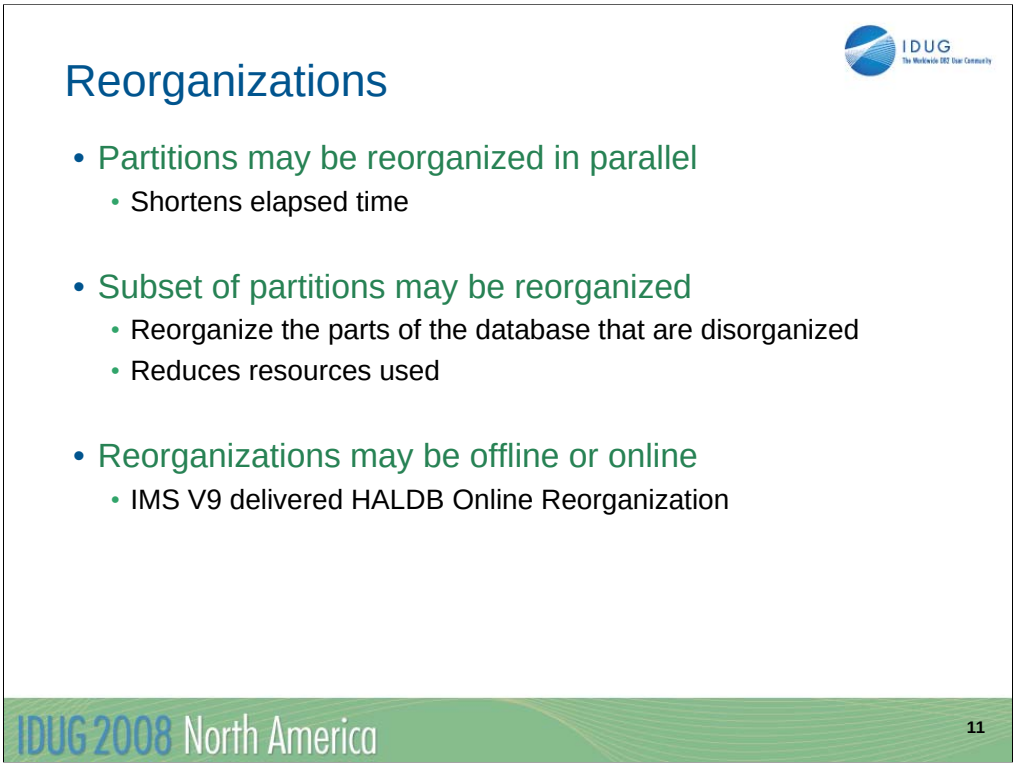
If a database has a secondary index, a timestamp recovery for a partition of the database usually requires a timestamp recovery for the secondary index. This, in turn, requires a timestamp recovery of the other partitions in the indexed database. Similar considerations apply to databases with logical relationships. If any partition is timestamp recovered, usually all partitions of the database and all partitions of its logically related databases must be recovered to the same time.



The techniques used for HALDB data sets with disaster recovery will depend on how disaster recovery is done.

If data mirroring is used, the HALDB database data sets, including ILDSs and PHIDAM indexes, will be covered by mirroring. They do not require special treatment.

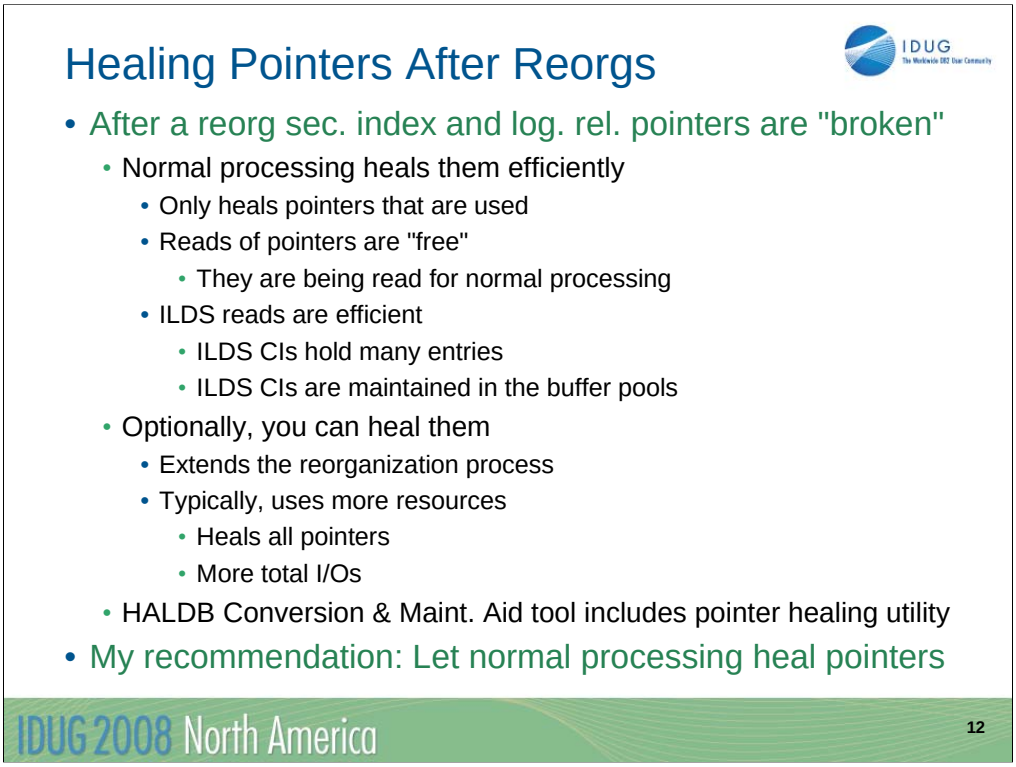
If image copies and logs are used, HALDB data sets are recovered like other IMS database data sets with the exception of ILDSs and PHIDAM indexes. These have to be rebuilt with DFSPREC0 or, possibly, IBM IMS Index Builder.



There are several alternatives for reorganizations of HALDB. Partitions may be reorganized in parallel to shorten the elapsed time. This is a major advantage of HALDB.

If activity is not evenly spread across the database, it may be desirable to reorganize only some partitions. Of course, these would be the partitions which are most disorganized. This would reduce the use of resources for reorganization.

With the introduction of HALDB Online Reorg in IMS V9, reorganizations may be either offline or online.



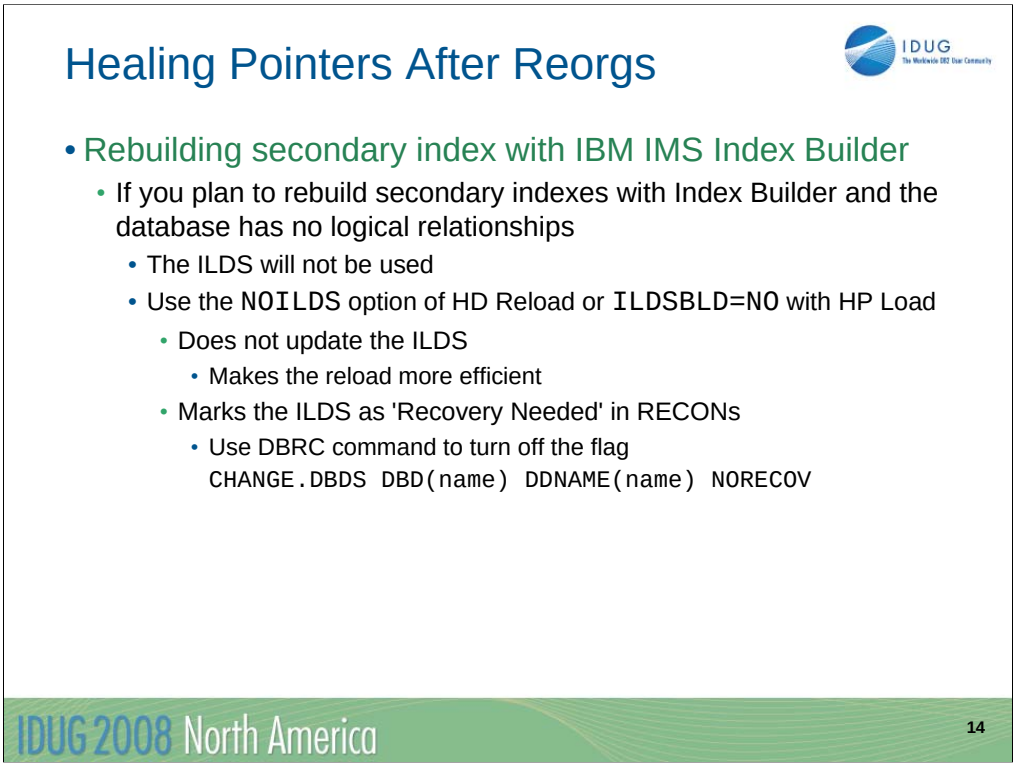
The self healing pointer process is used to correct or "heal" pointers after the segments to which they point have been moved by reorganizations. This process is generally very efficient. It only heals those pointers which are used. It does not waste resources updating pointers that are not used. There are many such pointers in most databases or indexes. Since the pointer is retrieved in any case, the healing process does not require any extra I/Os to read these blocks or CIs. The healing process gets the new data from the ILDS. ILDS entries (ILEs) are 50 byte records. CIs are much larger, so a CI will have many ILEs. These CIs use database buffer pools, just like other IMS database data sets. If they are heavily used, they will tend to be in the buffer pool. This will eliminate many reads for them.

Of course, you could explicitly heal the pointers. You can do this with any program which accesses the pointers using an update PROCOPT. The HALDB Conversion and Maintenance Aid tool includes a utility designed to heal pointers after an offline reorganization. In either case, this might require more resources. First, all pointers are healed, including those that are not used. Second, the pointers have to be read. This requires extra I/Os. My recommendation is to let normal processing heal the pointers. At least, let this be your default. If you find that this is not satisfactory, you can heal them explicitly.

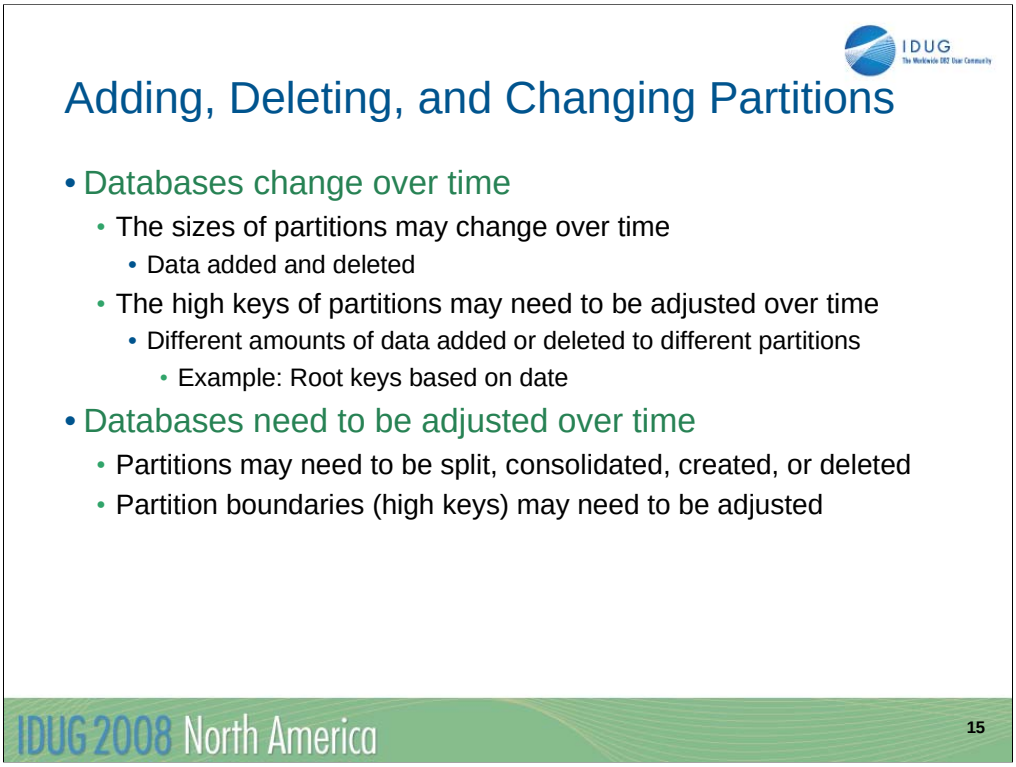
Healing Pointers After Reorgs

- Alternative process for secondary indexes
  - Secondary indexes do not have to be rebuilt after a reorganization of a HALDB database
    - But, secondary indexes may be rebuilt
  - Secondary indexes may be rebuilt with IBM IMS Index Builder
    - The rebuilt indexes are "organized"
      - Avoids possible need to reorganize secondary indexes
    - The pointers in the indexes are "healed"
      - ILDS is not used for secondary indexes when they are rebuilt by Index Builder
      - Pointers are "accurate" when rebuilt by Index Builder
        - Eliminates the need to heal the pointers

There is an alternative that you may want to consider when you reorganize a database which has secondary indexes. The secondary indexes do not have to be rebuilt, but you might want to consider rebuilding them with the IBM IMS Index Builder tool. IMS Index Builder builds the secondary indexes by writing the entries sequentially in load mode. This creates reorganized indexes. This avoids the possible need to reorganize the secondary indexes. This technique has a second advantage. The secondary index entries are accurate. That is, they are "healed". There will be no overhead to heal the pointers later. A third advantage is explained on the next page.



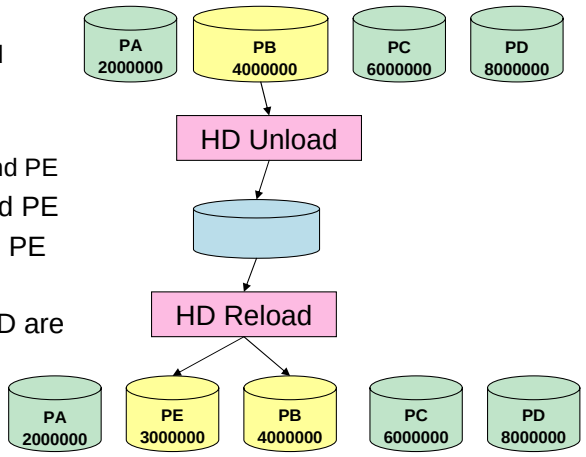
If you rebuild secondary indexes with Index Builder and the indexed database has no logical relationships, the ILDS will not be used. HD Reload and the High Performance Load tool allow you skip the updating of the ILDS. If you use Index Builder to rebuild all of the secondary indexes and there are no logical relationships, you should not update the ILDS. The NOILDS control statement for HD Reload and the ILDSBLD=NO parameter for HP Load should be specified. This saves the overhead of updating the ILDS. This control statement for HD Reload and this parameter for HP Load turn on the 'Recovery Needed' flag for the ILDS. You may use the DBRC command shown here to turn the flag off.



After you have used HALDB for some period, you will probably need to modify some database partitions due to changing database requirements. You may need to add partitions, deleted partitions, or change their boundaries. This is the natural result of the changes in the data in a database. An obvious example is a database whose root key is based on a date. New data is likely to be added with more recent dates. Deleted records are likely to be from less recent dates. Over time, new partitions may be needed for future dates. Partitions for older dates may be combined. Of course, there are many other situations which can lead to changing partition requirements.

Splitting a Partition

- If partition PB with high key 4000000 needs to be split
  - Unload partition PB
    - HD Unload or HP Unload
  - Define new partition PE
    - With high key 3000000
    - Sets PINIT flag for PB and PE
  - Initialize partitions PB and PE
  - Reload partitions PB and PE
    - HD Reload or HP Load
  - Partitions PA, PC, and PD are not affected



IDUG 2008 North America 16

This illustrates the requirements for splitting a partition. In this case, partition PB is very large. We want to split it into two partitions.

First, we unload partition PB. We may use the IMS HD Unload utility or the IBM HP Unload tool. Only partition PB is unloaded.

Second, we define a new partition. This is partition PE. The new partition has a high key greater than that for partition A and less than that for partition PB. The high key for PE is 3000000.

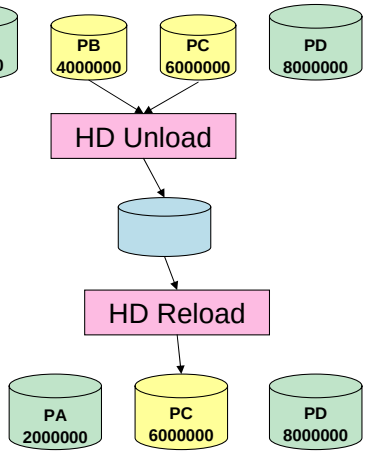
Third, we initialize partitions PB and PE. To do this we run either the HALDB Partition Data Set Initialization utility or the Database Prereorganization utility specifying this database. When we defined the new partition with a high key of 3000000, the 'partition initialization required' flag was automatically set for partitions PB and PE. IMS understands that records will potentially be moved from partition PB to partition PE. It sets the flags for these partitions.

Fourth, we run the reload using the output of the unload step. We may use either the IMS HD Reload utility of the IBM HP Load tool. This will load data into partitions PB and PE. This completes the process.

Partitions PA, PC, and PD are not affected by the change.

Combining Partitions

- If partitions PB and PC with high keys 4000000 and 6000000 need to be combined
  - Unload partitions PB and PC
    - HD Unload or HP Unload
  - Delete definition of partition PB
    - Sets PINIT flag for PC
  - Initialize partition PC
  - Reload partition PC
    - HD Reload or HP Load
  - Partitions PA and PD are not affected



Of course, you might need to combine some partitions. This example shows how you would combine partitions PB and PC. The process is similar to the split shown on the previous page.

First, you unload the two partitions to be combined.

Second, you delete the definition of the partition with the lower high key. This is partition PB. This sets the 'partition initialization required' flag for partition PC.

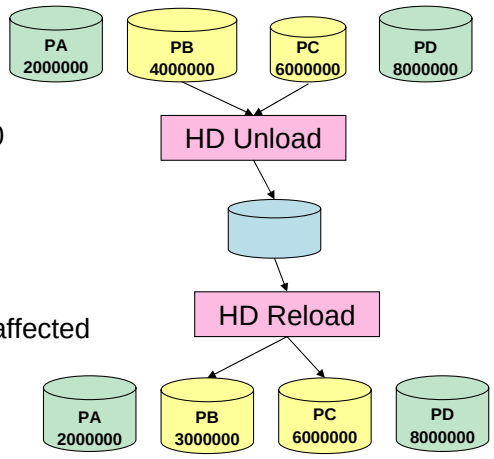
Third, you initialize partition PC.

Finally, you run reload which loads the data into partition PC.

Partitions PA and PD are not affected by the process.

Modifying Partition Boundaries

- If records need to be moved from partition PB to PC
  - Unload partitions PB and PC
    - HD Unload or HP Unload
  - Change high key for partition PB From 4000000 to 3000000
    - Sets PINIT flag for PB and PC
  - Initialize partitions PB and PC
  - Reload partitions PB and PC
    - HD Reload or HP Load
  - Partitions PA and PD are not affected



You also might need to adjust the sizes of some partitions. Over time some partitions might grow while others hold less data. This example show how you would move records between adjacent partitions. In this case, we want to move records from partition PB to partition PC. The process is similar other processes for adjusting partitions.

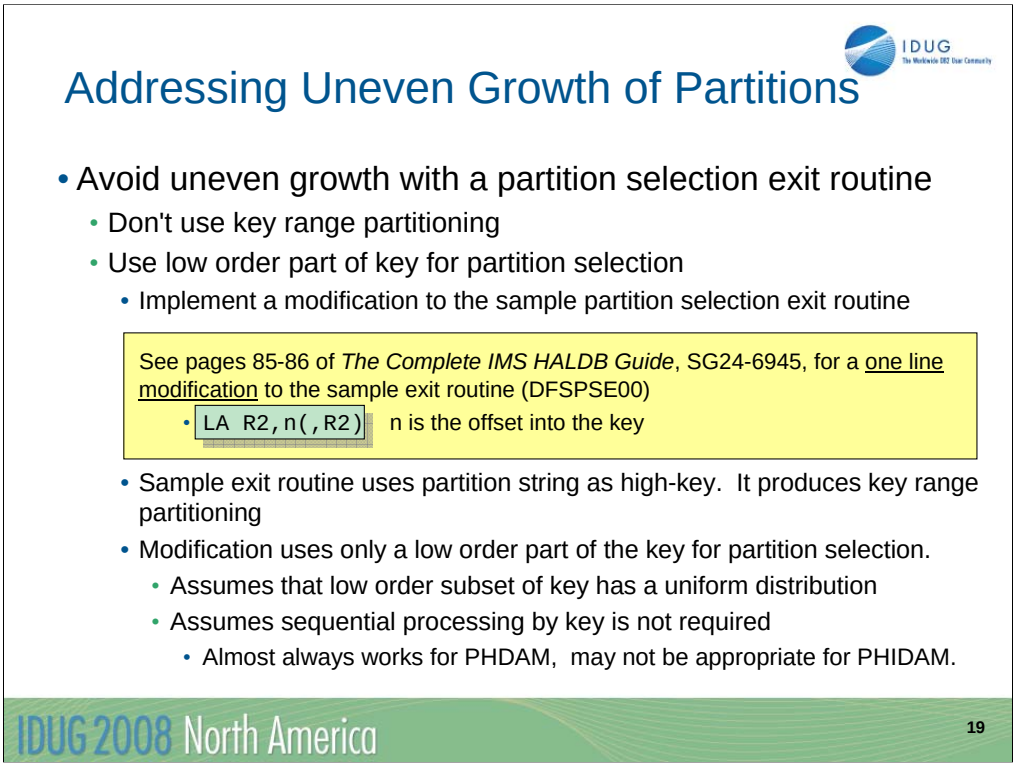
First, you unload the two partitions to be modified. These are partitions PB and PC.

Second, you change the high key for partition PB. Since this moves the assignment of some records from PB to PC, it sets the 'partition initialization required' flag for partitions PB and PC.

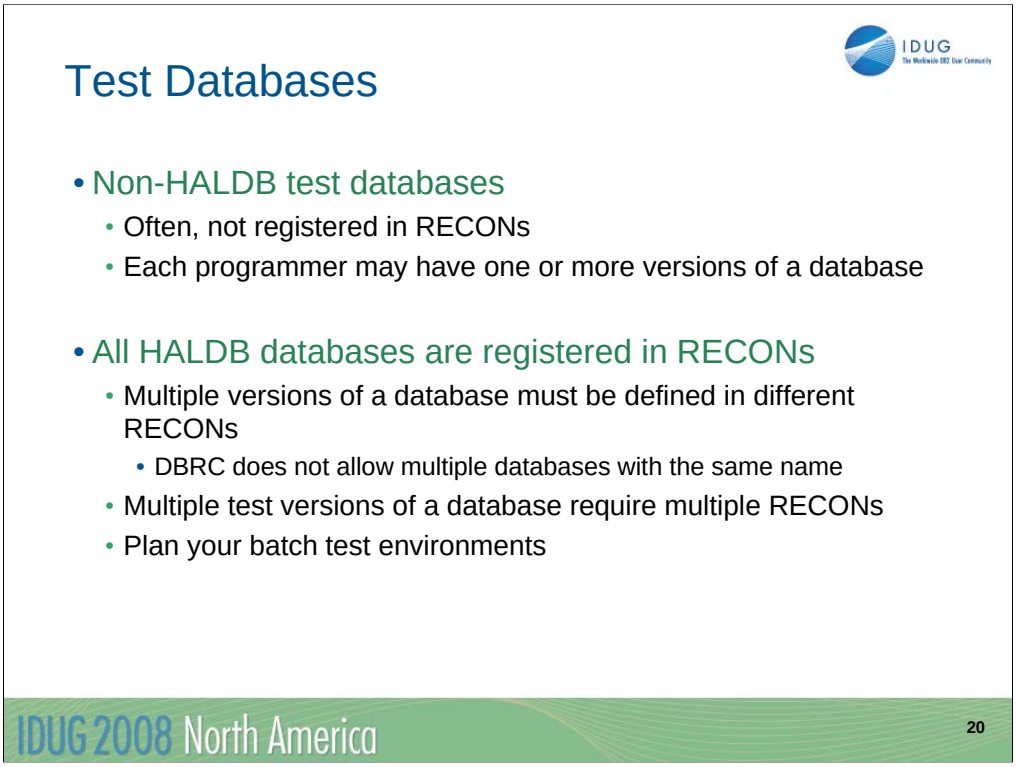
Third, you initialize partitions PB and PC.

Finally, you run reload which loads the data into partitions PB and PC.

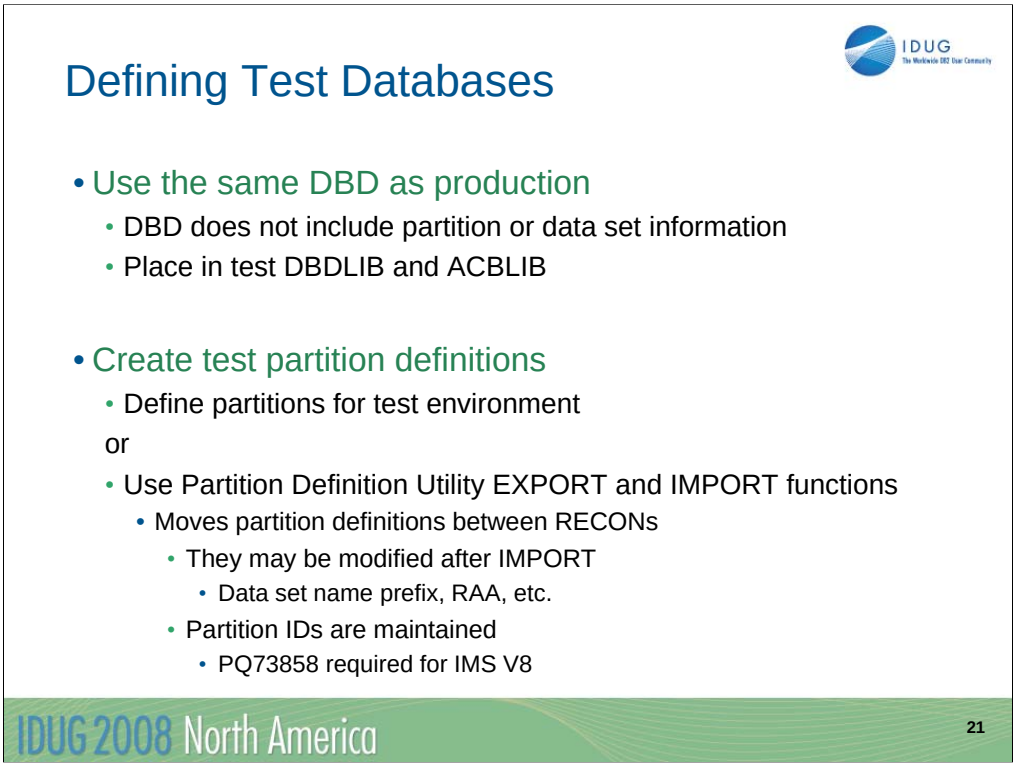
Partitions PA and PD are not affected by the process.



There is a technique that may be used to avoid uneven partition growth. If you use key range partitioning and have uneven growth, it is due to a change in the keys that are in use that effects the different partitions differently. The obvious way to avoid it is to assign keys to partitions so this does not happen. Usually, this can be done by using a partition selection exit routine. This is surprisingly easy to do. IMS supplies a sample partition selection exit routine, DFSPSE00, but it implements key range partitioning. The redbook shown here has an explanation of how to modify this exit routine to use a low order part of the key. It only requires a one line modification to the sample exit routine. You merely add a LA instruction to set the offset within the key. This causes the exit routine to selection a partition based on the this part of the key. Usually, you can find a part of the key that will produce a uniform distribution of records across the partitions. There is one warning about using this technique. It assumes that sequential processing by the key is not required. Since PHDAM does not place keys in sequence within a partition, having them out of sequence across partitions should not be a problem. This means that this technique almost always works with PHDAM. On the other hand, it will not work with PHIDAM if the keys must be in order for sequential processing.

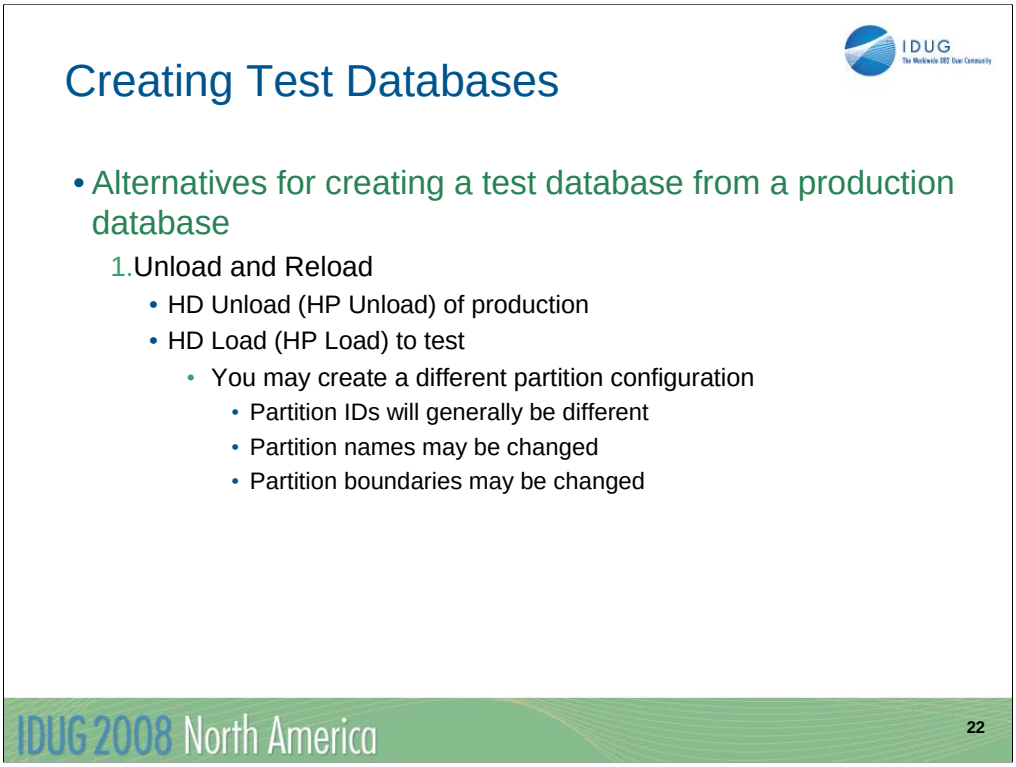


Testing of HALDB databases may be different from that for non-HALDB databases. Some users do not register test databases with DBRC. Since all HALDB databases must be registered, test HALDB databases must be registered. This means that different programmers cannot have different test databases with the same database name unless they also have different RECONs. These considerations must be remembered when planning test environments with HALDB.



When you create a test database you may use the same DBD as used in production. The DBD does not include any partition or data set information, so your test database may use the same DBD source but have a different partition structure and different DDNAMEs.

You may create partition definitions in two ways. First, you may use the Partition Definition Utility (PDU) or DBRC commands as you use them to create production partitions. Second, you may export the partition definitions from the production RECONs and import them to the test RECONs. After these definitions have been imported, they may be modified to meet your needs. When the listed APAR is applied for IMS V8, the partition IDs are maintained by the EXPORT and IMPORT process. This always occurs with IMS Versions 9 and 10. This is significant if you want to move a copy of a database data set to the test database. The partition ID is physically stored in the database, so a recovery to the test system using an image copy of the production system requires that the partitions have the same partition IDs.



There are several ways to create a test database from a production database.

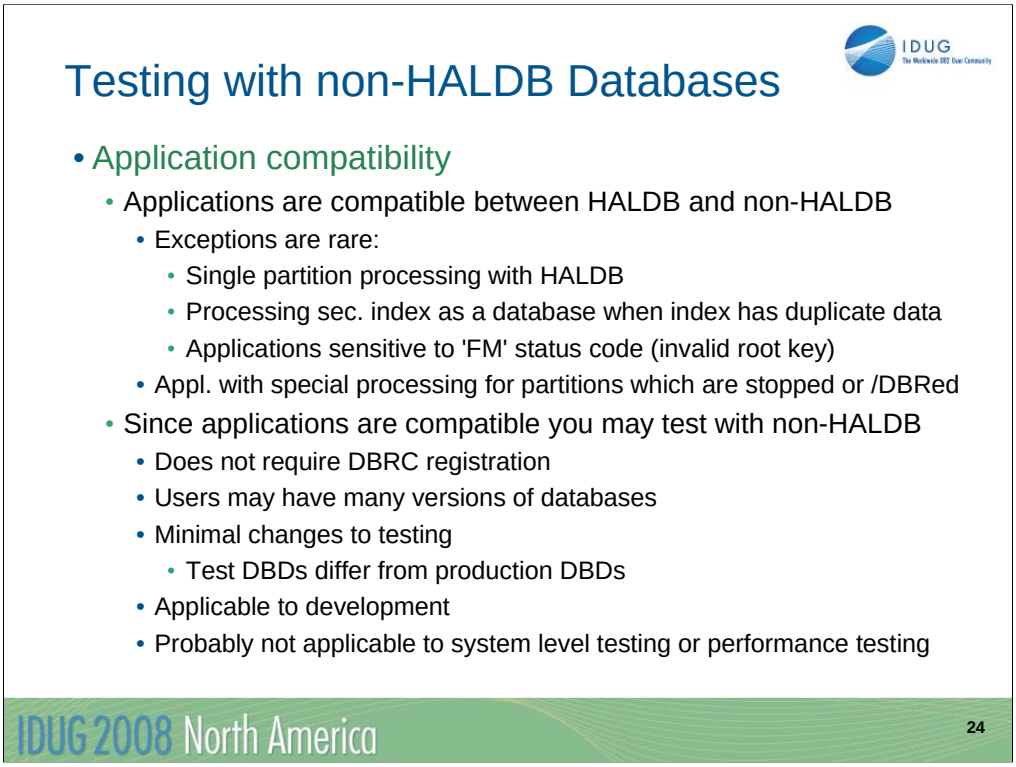
You can unload the production database and use the output as input to a reload of the test database. The partitions in the test database may differ from those in the production database. If you want only a subset of the production data in your test database, you can unload a subset of the production partitions.

Creating Test Databases

- Alternatives for creating a test database from a production database (continued)
  2. Image Copy and restore
     - Export and import partition definitions
       - Maintains partition IDs (requires APAR PQ73858 for V8)
     - Image copy production database data sets and restore to test
       - Partition IDs are stored in database data sets
     - Copy ILDS and PHIDAM primary index for each partition or use DFSPREC0
     - Change database data set names of test database
       - Change data set name prefixes
  3. Use application programs

You can image copy the production database data sets and restore them to the test database. Since the partition IDs are stored in the database, they must match the definitions in the RECONs. As mentioned before, an export and import of the definitions maintains these IDs. Of course, you will want to change the data set names. This is done by changing the data set name prefixes. The ILDS and PHIDAM indexes cannot be image copied. They can be copied by REPRO from the production system and restored with new names to the test system. Alternatively, you may build them by using the HALDB Index/ILDS Rebuild utility (DFSPREC0).

Of course, you can write application programs to load the test databases.



HALDB users do not necessarily have to do all of their testing with HALDB databases. Some testing can be done with non-HALDB full function databases. Application programs are not sensitive to using HALDB versus non-HALDB databases with just a few exceptions. The exceptions are:

•Execution with control statements that limit PCBs to a partition or set of partitions

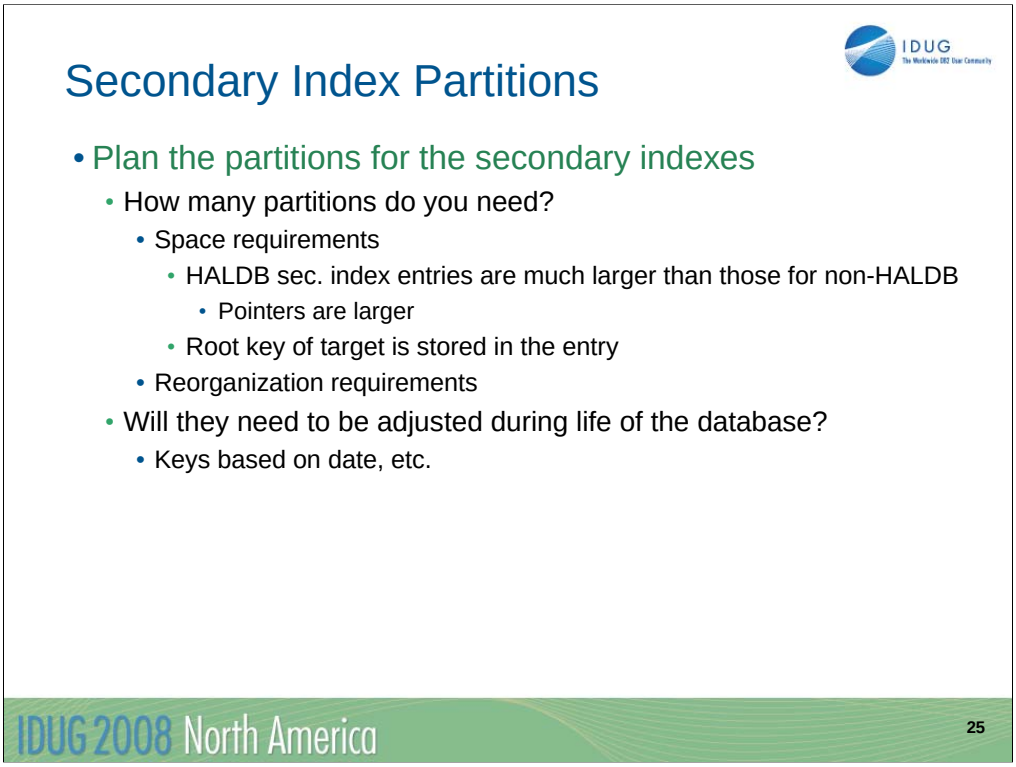
•Processing a secondary index as a database when the index has duplicate data

•Applications that are sensitive to the 'FM' status code. This status code is returned when an application attempts to insert a root which is invalid in the database because there is no partition which includes it in its key range or the partition selection exit routine rejects the key.

Applications that have special processing for unavailable data due to individual partitions being stopped or unavailable.

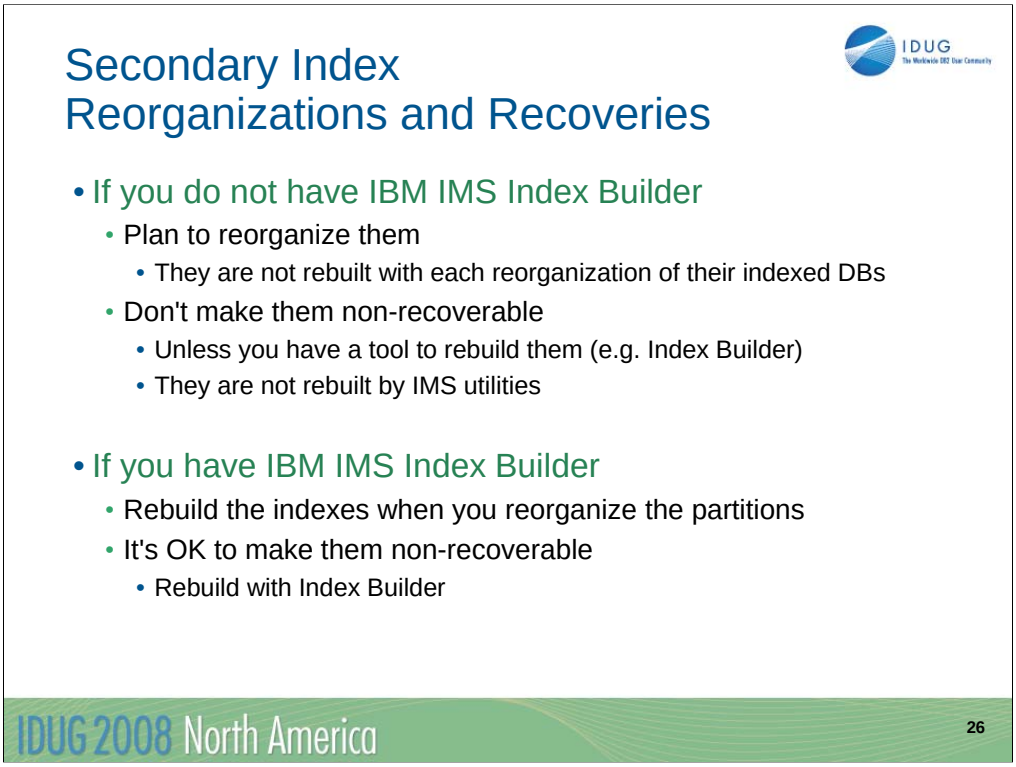
Since applications are generally unaware whether the database is HALDB or not, you may be able to simply some testing by using non-HALDB databases. This eliminates the requirements for DBRC registration and allows you to have multiple versions of the database without requiring multiple RECONs. This is especially attractive if this is the way you test with non-HALDB databases.

Of course, you will want to do some testing with databases defined as HALDB. Testing with non-HALDB databases will probably be limited to application development and unit testing. System and performance testing will more likely require actual HALDB databases.



Secondary indexes are partitioned also. When creating a secondary index you must decide how many partitions you want. As with the databases, this will depend on factors such as the size and number of records. HALDB secondary index records tend to be much larger than non-HALDB secondary index records. HALDB uses a 28-byte extended pointer set, not a 4-byte RBA, for a pointer. Also, the root key of the target is stored in the secondary index record. These make the index records larger. Like other databases, the size of the partitions will affect the time required to reorganize them.

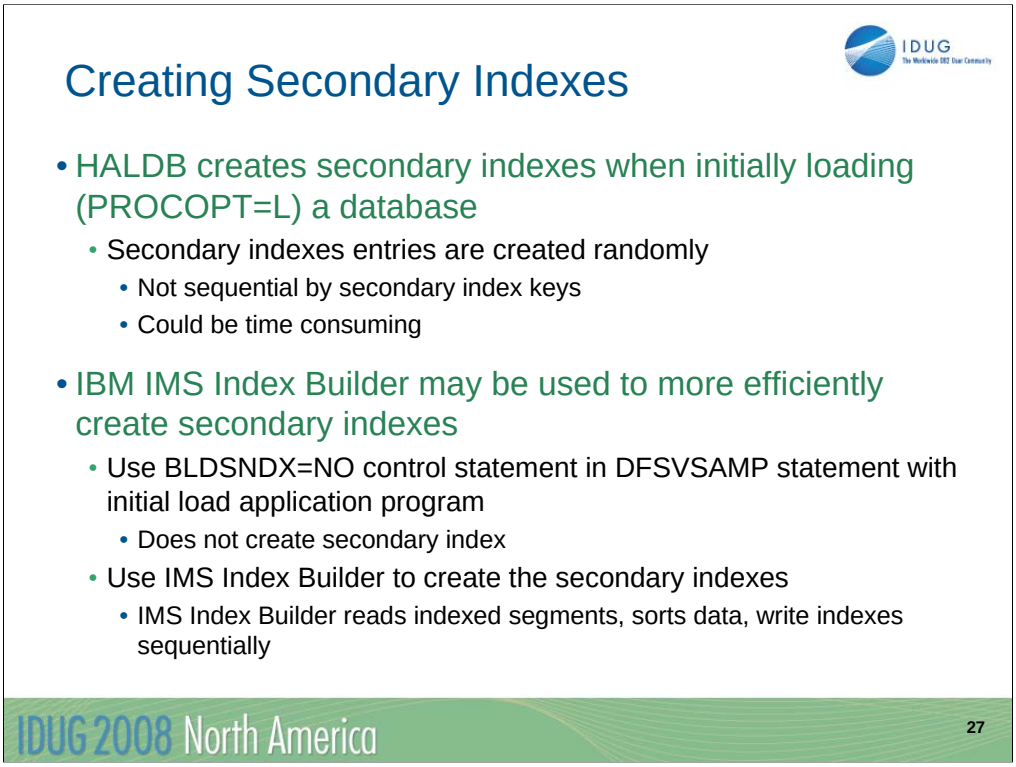
As databases change, their partitioning requirements may change. This also applies to secondary indexes. They should be monitored as other databases are monitored.



Most HALDB secondary indexes need to be reorganized occasionally. Since a reorganization of the indexed database does not result in the rebuilding of the secondary index, the secondary index may become disorganized over time if it is not reorganized.

Many installations make non-HALDB secondary indexes non-recoverable. This eliminates logging for changes to the secondary indexes. If they must be recovered, they may be rebuilt from the indexed databases using standard IMS utilities. This is not true for HALDB secondary indexes. They cannot be rebuilt using standard IMS utilities. They can be rebuilt by tools, such as the IBM IMS Index Builder. You should have such a tool if you make a HALDB secondary index non-recoverable.

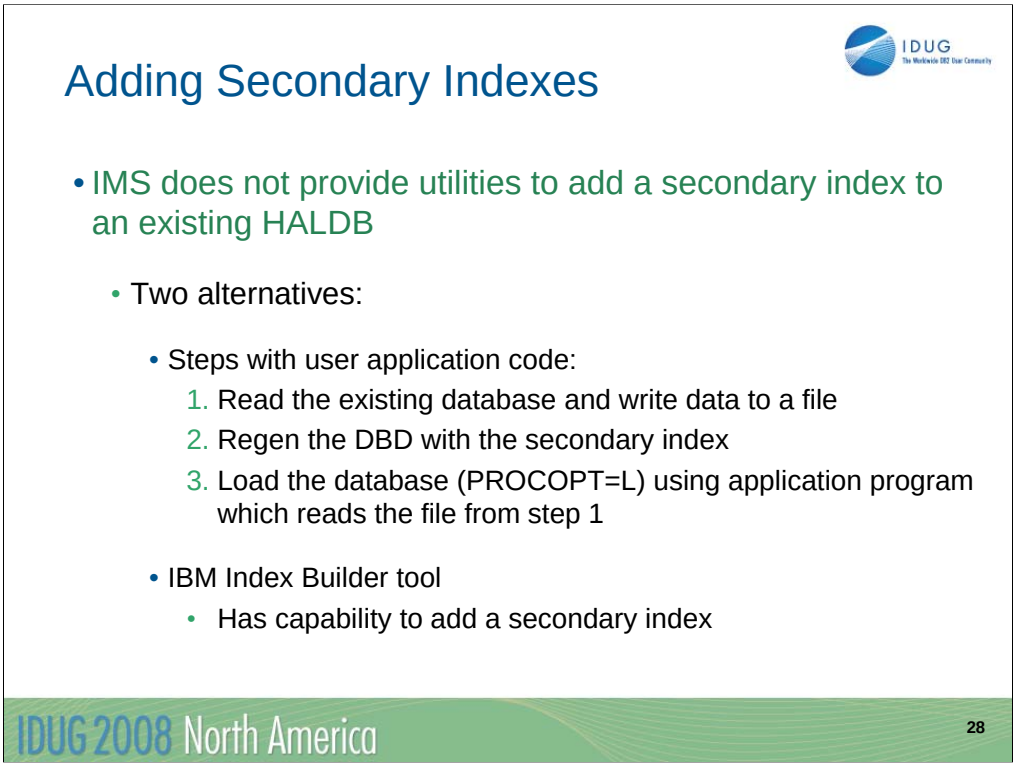
If you have a tool like IBM IMS Index Builder you may rebuild the secondary indexes when you reorganize the partitions of a database. This would probably eliminate the need to reorganize the secondary indexes. The tool would also allow you to define the secondary indexes as non-recoverable since the tool may be used to rebuild the secondary indexes without image copies or logs.



When you initially load a HALDB database with secondary indexes, the indexes are built during the load. This is not like non-HALDB databases. Non-HALDB database loads create a work file which is processed by the Prefix Resolution, HISAM Unload, and HISAM Reload utilities to create the secondary indexes. HALDB does not use these work files or utilities. Since HALDB secondary index entries are created when the indexed segments are inserted, the updates to the secondary index are typically random. These random inserts may be time consuming. They may have a significant impact on the elapsed time of the load job.

If you use a tool, such as the IBM IMS Index Builder tool, you may be able to load databases with secondary indexes more quickly. If you include a BLDSNDX=NO control statement in your DFSVSAMP data set, the secondary indexes are not created. This avoids the random inserts. Of course, you need to create the secondary indexes. You can use the IBM IMS Index Builder tool to build the secondary indexes. It reads the indexed segments, sorts the data needed to create the index entries, and writes the secondary indexes sequentially. This can substantially reduce the time required to create the database with its secondary indexes.

The BLDSNDX control statement was added by APAR PQ56463 for IMS V8. It is included in IMS V9 and IMS 10.

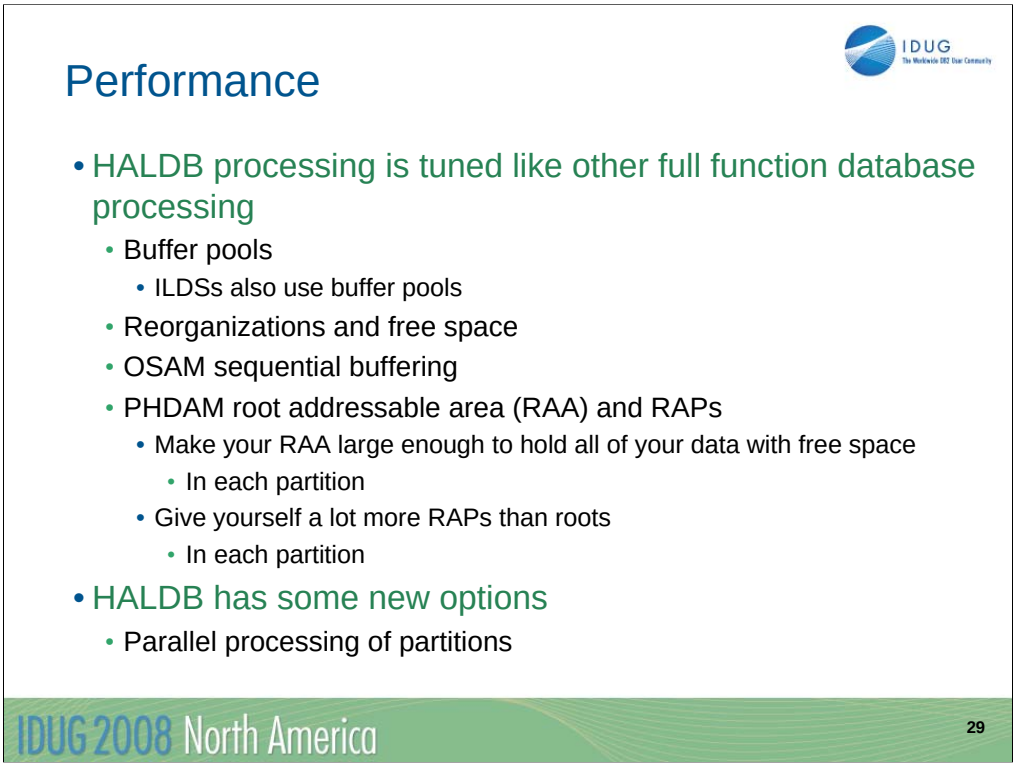


The utilities supplied with IMS do not provide a way to add a secondary index to an existing HALDB database. Secondary indexes are added to non-HALDB databases by unloading the database, changing the DBD to include a secondary index, reloading the database, and then building the secondary index from the work file produced by reload. HALDB does not use work files, so this technique cannot be used to add a secondary index to a HALDB database.

There are two alternatives for adding a secondary index.

The first alternative uses application programs. Step one unloads the existing database to a file. Step two modifies the existing DBD to include the secondary index and creates a DBD for the secondary index. Step three uses an application program to load the database with PROCOPT=L. The secondary index is created when the database is loaded.

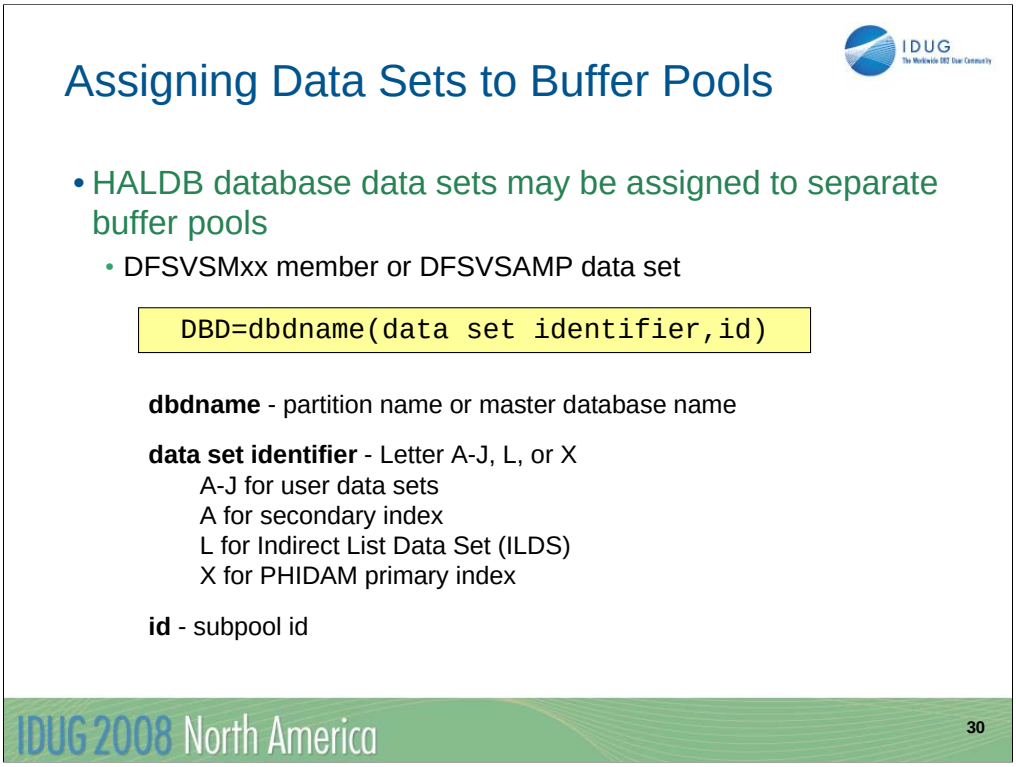
The second alternative uses the IBM IMS Index Builder tool. This product can add a secondary index to an existing database without unloading or reloading the database.



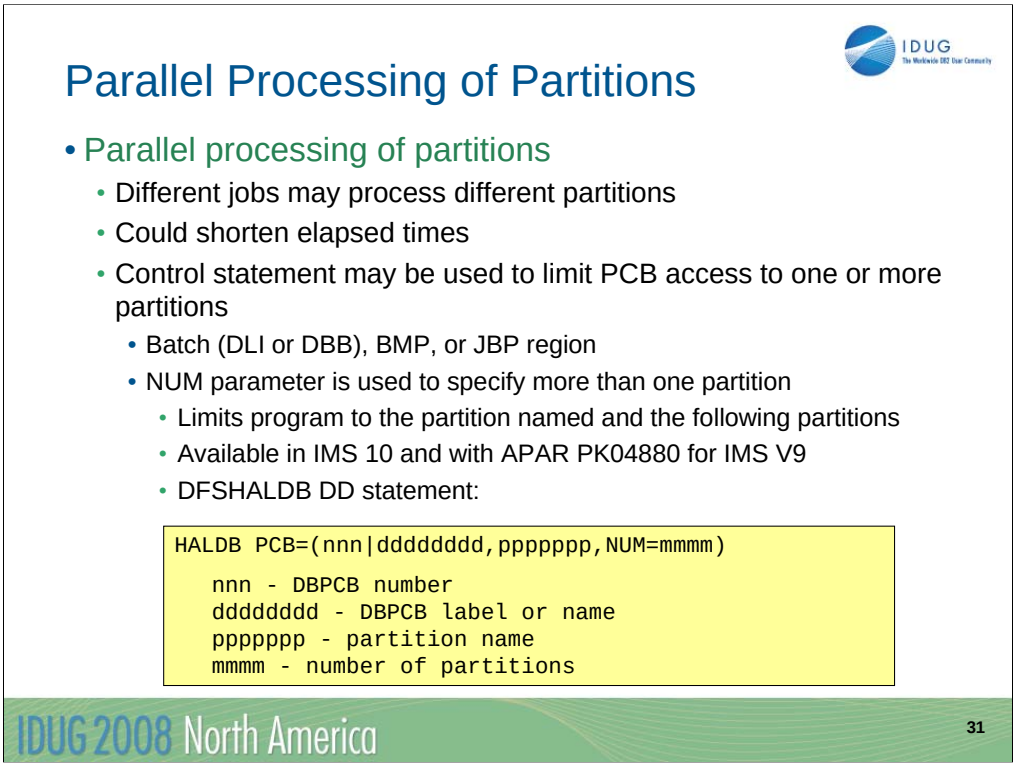
In general, HALDB databases are tuned like other full function databases. Data sets may be assigned to buffer pools. ILDSs are database data sets. They also use database buffer pools. The assignment of data sets to buffer pools is shown on the next page. Reorganizations and free space have the same importance for HALDB as they do for other full function databases. OSAM sequential buffering is supported and should be used for sequential processes.

Each PHDAM partition has its own root addressable area. As with HDAM, the RAA should be large enough to hold the data and free space. Also, you should have enough RAPs to avoid long synonym chains for root segments.

Of course, HALDB has some new performance opportunities. You have the capability to easily process different partitions in parallel. This may significantly reduce the elapsed time for some processing.



The assignment of data sets to buffer pools is similar to that for non-HALDB databases. The only difference is that the data set identifier on the DBD statement for DFSVSMxx or DFSVSAMP is a letter, not a number, and the dbdname may be either the database name or the partition name. HALDB and non-HALDB data sets may share the same buffer pool.



IMS provides a capability to limit a PCB to one or more partitions of a database. This is done with the HALDB control statement in the DFSHALDB DD data set. The control statement specifies a PCB, a partition, and possibly a number of partitions. When the control statement is used, the PCB only has access to the specified partition(s). An unqualified call without previous position will return the first segment in the names partition. A GN call which reaches the end of the partition or partitions receives a 'GB' status code. APAR PK04880 for IMS V9 added the capability to specify the NUM parameter. This specifies the number of partitions that may be processed with the PCB. For example, if you specify NUM=4, the PCB will be limited to 4 partitions beginning with the specified partition.

More Information on HALDB

- IMS Product Publications:
  - *IMS V9 Administration Guide: Database Manager*
  - *IMS 10 Database Administration Guide*
  - *IMS V9 Utilities Reference: System*
  - *IMS 10 System Utilities Reference*
  - *IMS V9 Utilities Reference: Database and Transaction Manager*
  - *IMS 10 Database Utilities Reference*
- Redbook:
  - *The Complete IMS HALDB Guide*, SG24-6945
    - Written in 2003, but still valuable
- TechDocs
  - www.ibm.com/support/techdocs
    - Many presentations, white papers, and flashes on IMS and other products

More Information on HALDB

- Presentations on TechDocs:
  - *IMS HALDB Database Administration*
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS842
  - *IMS High Availability Large Database (HALDB)*
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS185
  - *Migrating to IMS HALDB (High Availability Large Database)*
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS693
  - *Application Design and Programming with HALDB*
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS490
  - *IMS Version 9 HALDB Online Reorganization*
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1254

Presentations include text to explain the slides.

More Information on HALDB

- White Papers on TechDocs:
  - *IMS Full Function Database Design Guidelines*
    - Provides guidance for all full function databases including HALDB
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100731
  - *IMS HALDB Reorganization Number Verification*
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100733
  - *Converting IMS Logically Related Database DBDs to HALDB*
    - http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100763

HALDB Database Administration

- Recovery
  - Similar to other database data sets
    - Special considerations for ILDS and PHIDAM index
- Reorganizations
  - Alternatives including healing pointers
- Modifying partitions
  - Unload affected partitions, redefine, and reload
- Test databases
  - HALDB requires DBRC registration
- Secondary indexes
  - Alternatives including use of IBM IMS Index Builder
- Performance
  - Full function database with some new options

J09

IMS HALDB Database Administration

Rich Lewis
IBM
rzlewis@us.ibm.com