IMS HALDB Reorganization Number Verification

Rich Lewis
IMS Advanced Technical Support
IBM Americas

March 2006

HALDB Partition Number Verification

Recent IMS maintenance has added a capability for IMS to verify the reorganization numbers of HALDB partitions. This enhancement has improved the integrity of HALDB databases. The maintenance was delivered as a result of APARs PQ97357 for IMS Version 9 and PQ97356 for IMS Version 8. This paper explains the need for the function, how it operates, and how you can implement it. It also provides guidance on avoiding the potential database integrity exposure before you apply this maintenance. Finally, it provides background information about how IMS uses reorganization numbers with HALDB partitions.

Without the HALDB partition number verification function, HALDB databases can be corrupted by invalid reorganizations. This function eliminates this exposure.

If you are unfamiliar with IMS's use of partition reorganization numbers and indirect list keys (ILKs), you should read the Partition Reorganization Number Background section at the end of this document before proceeding.

The Potential Problem

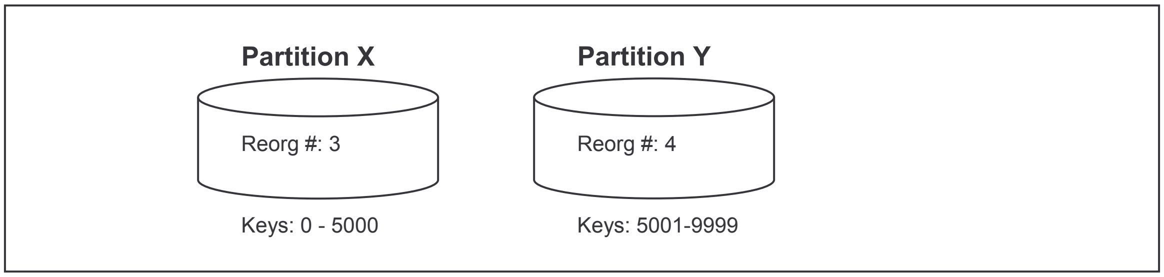
Without this enhancement reorganization numbers can be regressed. That is, a reorganization might not increment a reorganization number. It could leave the reorganization number at its old value or lower the value. After the reorganization a new segment inserted in the partition could have the same indirect list key (ILK) as a previously inserted segment. This is a potential integrity problem if the newly inserted segment is the target of a secondary index or logical relationship. Two segments could have the same ILK. These segments would have only one Indirect List Entry (ILE) in the Indirect List Data Set (ILDS). When the pointers to these two segments are updated, they could point to the same target. Half of the time this would be the wrong segment.

The exposure to this problem occurs when partition boundaries are changed and the partitions have been reorganized a different number of times.

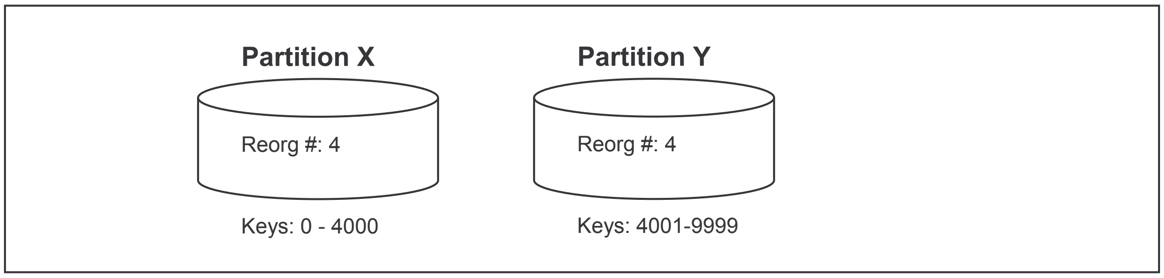
Without this enhancement IMS keeps the reorganization number for a partition only in the partition data set. It is not kept in the RECONs. When the partition is unloaded, the reorganization number is read and placed in the prefix of each segment in the unload data set. When reload inserts the first segment in a partition, it takes the reorganization number from the segment's prefix in the unload data set, increments it, and writes it to the partition data set. This process may not increment the reorganization number when partition boundaries are changed.

The following is an example of a situation where the reorganization number of a partition is not incremented.

A database has two partitions, partition X and partition Y. Partition X contains keys 0 through 5000. Partition Y contains keys 5001 through 9999. The reorganization number for partition X is 3. The number for Y is 4.



These partitions are reorganized and their key ranges are changed. Both are unloaded. The partition definitions are changed so that the key range for partition X is 0-4000 and the key range for partition Y is 4001-9999. When the reload is done, the first segment inserted into partition Y was formerly in partition X. The reorganization number in its prefix is 3. This number is incremented to 4 and becomes the new reorganization number for partition Y. Unfortunately this is the same as the old reorganization number for partition Y. The following diagram illustrates the situation after the reorganization.



As you can see, the reorganization number for partition Y has not been incremented. Its value remains 4.

The Solution

APARs PQ97357 for IMS Version 9 and PQ97356 for IMS Version 8 address this problem. With this maintenance IMS stores the reorganization number for a partition in the RECON partition database record. This number is incremented by reorganizations. This eliminates restrictions on changing partitions with reorganizations. The number in the RECON record is also verified against the number stored in the partition data set. This provides additional integrity checking.

The maintenance causes the following actions to occur:

- The reorganization numbers in the RECON partition database records are initially set to zero.
- Non-load mode update programs for a partition read the number from the RECON record. If it is a lower value than the value in the data set, the value in the data set is written in the RECON record. When implementing this maintenance, this typically will be the action that makes the RECON record value non-zero.
- HD Unload reads the reorganization number from the data set and writes it to the RECON record. It does not update the reorganization number in the partition data set. It also writes the reorganization number in the prefixes of the segments in the unload data set.

- HD Reload increments the value in the RECON record and writes it to the partition data set and the RECON record. It ignores the reorganization numbers in the unload data set.
- Partition initialization reads the number from the RECON record, increments the value, and stores it in the partition data set. It does not store the incremented value in the RECON record. This ensures that partition initialization of an existing partition does not regress its reorganization number. Partition initialization is a function of the HALDB Partition Data Set Initialization utility (DFSUPNT0) and the Database Prereorganization utility (DFSURPR0).
- Initial load sets the reorganization number to one in both the RECON record and the partition data set.

These actions ensure that reorganizations always increase partition reorganization values and that they are not regressed by other activities.

Implementing the Solution

The changes in these APARs will only take effect when you set a flag in the RECONs. Do not set the flag before you apply the maintenance to all SDFSRESL data sets (RESLIBs) which you use with the RECONs. The MINVERS value must be 8.1 or higher to set the flag. You set the flag with the following DBRC command:

```
CHANGE.RECON REORGV
```

A listing of the RECON header shows the current status of the flag as follows:

```
REORG NUMBER VERIFICATION = YES|NO
```

After reorganization number verification is implemented, you may receive some new messages. Message DSP1108I is issued when the reorganization number for a partition is incremented. The text of the message is:

```
DSP1108I REORG# CHANGED FROM xxxxx TO yyyyy FOR DATABASE zzzzzzz
```

In this message zzzzzzz is the partition name, not the database name.

Messages After Timestamp Recoveries

If you do a timestamp recovery to a time before the last reorganization, you must restore all of the data sets to the same time. This will require a rebuild of the ILDS. You may do this with the Index/ILDS Rebuild utility (DFSPREC0). These timestamp recoveries will regress the reorganization number in the partition data set to a value lower than that in the RECON record. This is OK. IMS uses the reorganization number in the RECON record. IMS updates the value in the data set when it first opens the data set for update. It then issues the DFS3280I message:

```
DFS3280I REORG# UPDATED FOR PARTITION yyyyyyy
```

If the partition is opened for input, IMS cannot update the reorganization number in the data set and issues the DFS3282W message. The text of the message is:

```
DFS3282W REORG# NEEDS UPDATE FOR PARTITION yyyyyyy
```

This is not a problem. IMS will use the reorganization number value from the RECONs. The number will be updated when the data set is opened for update.

IMS Tools

The IBM IMS tools, such as High Performance Unload, High Performance Load, Online Reorganization Facility, and the HALDB Conversion and Maintenance Aid do not require any maintenance for support of the new reorganization number verification function. If you use IMS tools from other vendors, you should check with your vendor to see if they require maintenance for support of the new function.

Warning If You Have Not Implemented the Solution

We highly recommend that you apply this maintenance. If you have not applied this solution, you should be careful when changing partition assignments. There is a potential data integrity problem. The exposure only occurs when the following two conditions are met:

1. Partition boundaries are changed
2. The partitions have been reorganized a different number of times

In addition, the problem occurs only when the first segment inserted into a partition by reload comes from a partition with a lower reorganization number.

The exposure does not exist when adding new partitions.

What can you do when the conditions exist and you want to change the partition boundaries? Instead of changing the boundaries on the existing partitions, you can delete them and create new partitions. The steps would be:

1. Unload the affected partitions.
2. Disable or delete the partition definitions for the affected partitions.
3. Define new partitions with new key ranges. If you delete the old partition definitions, these partitions may have the same names as the deleted partitions.
4. Reload the data.

When you define new partitions, they get new partition ID numbers. These numbers are used in the ILKs. So they cannot be duplicates of the ILKs for any previously existing segments.

Partition Reorganization Number Background

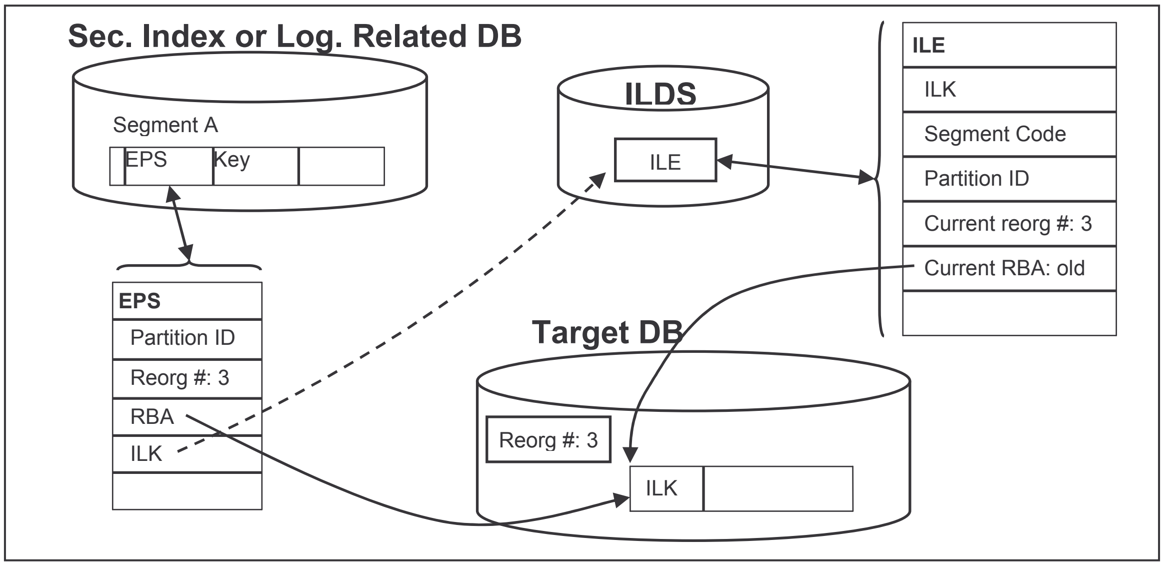
This section explains how partition reorganization numbers are used and maintained by IMS.

HALDB maintains a reorganization number for each partition in a PHDAM or PHIDAM database. The number is stored in the first block (or CI) of the first data set in the partition. It is updated when the partition is reorganized. This number is used as part of the self-healing pointer process. This process updates secondary index and logical relationship pointers after the partition has been reorganized.

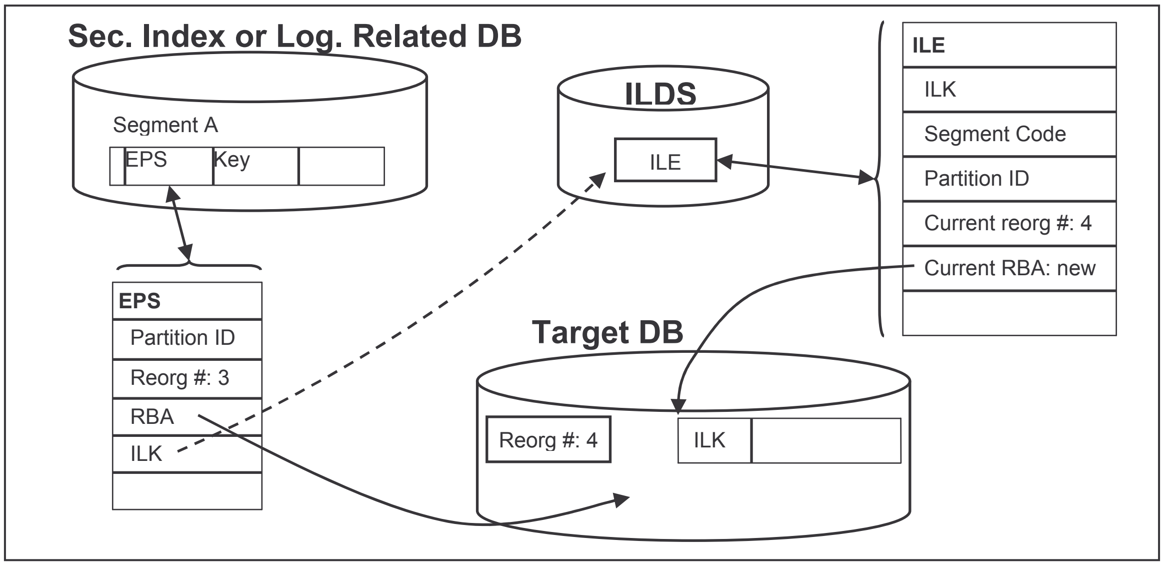
Secondary index and logical relationship pointers are stored in the extended pointer set (EPS) in the pointer segment. The reorganization number of the target partition is also stored there. This number indicates that the pointer was last updated when the target partition had the corresponding reorganization number. When these pointers are used, IMS compares the reorganization number in the EPS to the number in the target partition. If they match, IMS "knows" that the pointer is accurate. If the numbers do not match, IMS "knows" that a reorganization has been done since the pointer was last updated. IMS looks up the new location of the target segment in the Indirect List Data Set (ILDS) for the partition.

The key of the entry in the ILDS is the indirect list key (ILK). The ILK is associated with the target segment. The ILK is created when the segment is created and stored in the segment prefix. The ILK is built from three values. They are the relative byte address (RBA) of the segment when it was created, the partition ID number of the partition in which is was originally stored, and the reorganization number of the partition when the segment was created. The ILK value for a segment never changes after it is created. It is not changed by reorganizations. Since two segments cannot have the same RBA in the same partition at the same time, ILKs are unique across a database. At least, they should be.

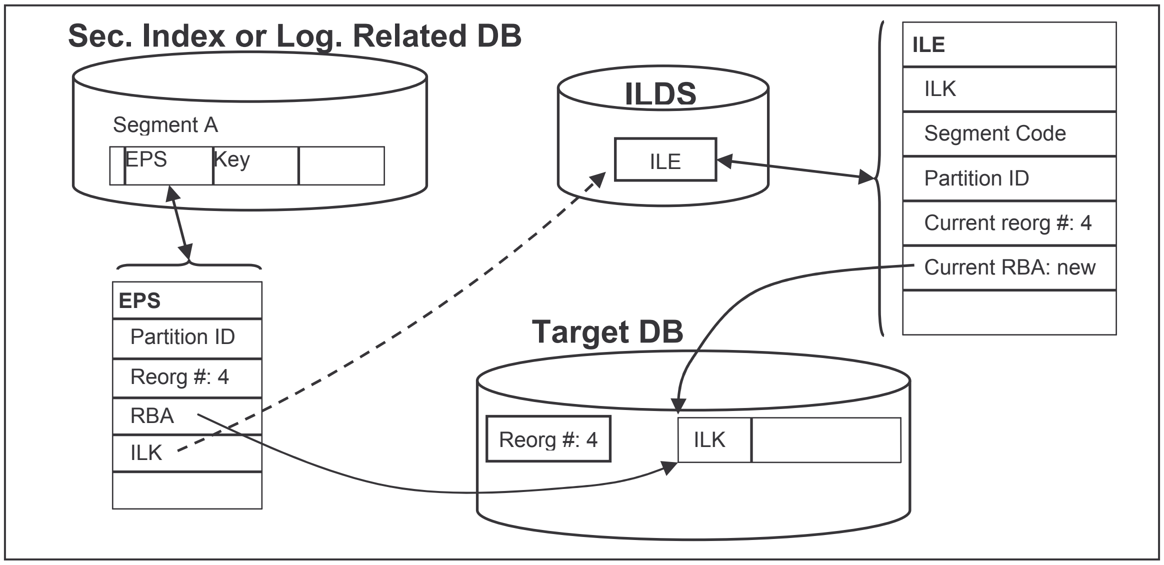
The following diagrams illustrate how pointers, reorganization numbers, and ILKs are used. In the first diagram the secondary index or logical relationship pointer has been updated after the last reorganization. That is, the "broken" pointer has been "healed." The reorganization number values in the partition data set and in the ILE are 3. The value of the reorganization number in the EPS of the pointer segment is also 3. The RBA in the EPS points to the current location of the target segment.



The following diagram illustrates the situation after a reorganization. The reorganization has moved the target segment, updated its ILE in the ILDS, and incremented the reorganization number in the partition data set. The reorganization number in the partition data set is 4. The reorganization number in the EPS is still 3. This indicates that the pointer in the EPS is inaccurate. IMS will have to look up the new RBA for the target segment. It uses the ILK to find the appropriate entry in the ILDS. This entry contains the new RBA for the segment.



When IMS looks up the new RBA for the target segment, it also updates the RBA field in the EPS with the new RBA value and the new reorganization number. This is the self-healing process. The following diagram illustrates the situation after this process.


Sec. Index or Log. Related DB
Segment A
EPS
Key
ILDS
ILE
ILE
ILK
Segment Code
Partition ID
Current reorg #: 4
Current RBA: new
EPS
Partition ID
Reorg #: 4
RBA
ILK
Target DB
Reorg #: 4
ILK