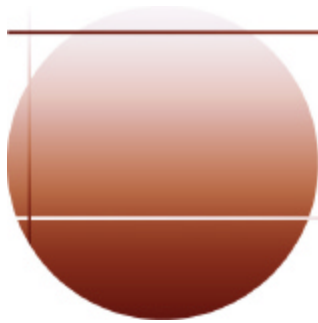


E81

Migrating to HALDB

Rich Lewis



IMS

technical conference

Las Vegas, NV

September 15 - September 18, 2003



Abstract and Trademarks

Abstract:

High Availability Large Database (HALDB) is a new capability introduced with IMS V7. Other sessions at this conference describe how HALDB allows a database to grow to many terabytes. HALDB databases also provide increased availability by dramatically shortening reorganization times and allowing parallel processing of multiple partitions of a database. So, how do we migrate an existing IMS database to HALDB? This session describes the process and the options for doing this migration. Migrations of databases that were partitioned by other means are included.

Trademarks:

The following are trademarks of International Business Machines, Inc.: IMS, IMS/ESA[®], IBM[®]

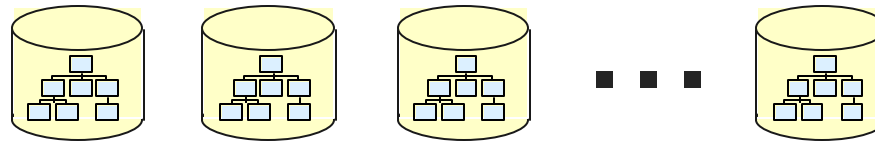
The following are trademarks of NEON Systems, Inc.: Partitioned Database Facility, PDF, NEON Systems, Inc.

Those trademarks followed by (®) are registered trademarks in the United States.

A version of this presentation which includes extensive notes is available at:
<http://www.ibm.com/support/techdocs/atmastr.nsf/WebIndex/PRS693>

Redbook: [The Complete IMS HALDB Guide, All You Need to Know to Manage HALDBs, SG24-6945](#), has much more information about HALDB.

HALDB Overview



▲ Databases are partitioned

- Up to 1001 partitions per database
- Partitions have up to 10 data set groups
- Partitions may be allocated, authorized, and reorganized independently

▲ Database Types

- Partitioned HDAM (PHDAM)
- Partitioned HIDAM (PHIDAM)
- Partitioned Secondary Index (PSINDEX)

▲ Self healing pointers

- Reorganization of partition does not require changes to secondary indexes or logically related databases which point to it



HALDB Overview - Pointers

▲ Logical child segments and secondary index segments include:

- Key of target
 - Key of target's root for secondary indexes
 - Logical Parent's concatenated key of for logical relationships
 - ▶ Used to determine partition in which target resides

- Extended Pointer Set (EPS)
 - ▶ Reorganization number of target partition when RBA pointer was accurate
 - Used to determine if RBA pointer is still accurate

 - ▶ RBA of target when last known

 - ▶ Indirect list key (key of Indirect List Entry) for target segment
 - Used when RBA pointer is not still accurate

▲ Other pointers are unchanged from non-HALDB databases



HALDB Overview - ILDS

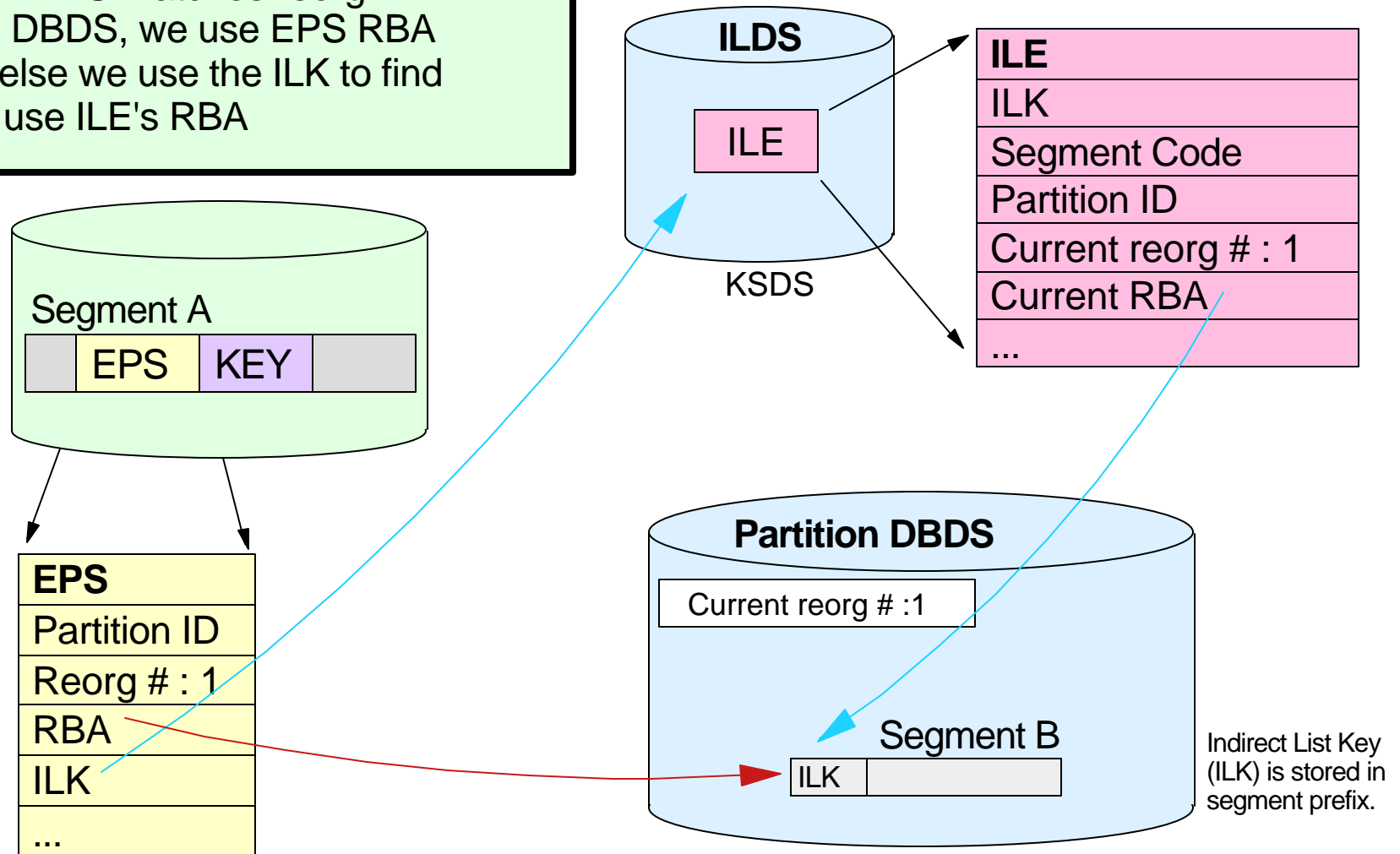
- ▲ **An Indirect List Data Set (ILDS) is associated with each partition**
 - One KSDS per partition

- ▲ **ILDS contains accurate RBA pointer to each secondary index or logical relationship target**
 - Entries are created or updated by reorganization
 - Key of entry is ILK (indirect list key) associated with target segment
 - Segments which are not targets of sec. ind. or log. rel. do not have entries

- ▲ **ILDS is used when reorganization number in logical relationship or secondary index pointer is "out of date"**
 - "Out of date" when reorg. number does not match partition's reorg. number
 - "Out of date" indicates that the pointer has not been healed since last reorg.

HALDB Overview - Using the EPS and ILE

If Partition ID in EPS is correct and reorg # in EPS matches reorg # in Partition DBDS, we use EPS RBA pointer, else we use the ILK to find ILE and use ILE's RBA





Migrating Databases to HALDB

▲ Migration Overview

- DBDGEN for HALDB

- Partition definitions for HALDB
 - ▶ Uses Partition Definition Utility

- HALDB database partitions are initialized with Prereorg utility

- Non-HALDB database is unloaded with HD Unload
 - ▶ Uses Non-HALDB DBD
 - ▶ Creates unload data set

- HALDB database is created with HD Reload
 - ▶ Uses HALDB DBD and RECONS
 - ▶ Created from unload data set



Migrating Databases to HALDB

▲ Considerations:

- All logically related databases must be migrated concurrently
 - ▶ No logical relationships between HALDB and non-HALDB databases
 - HALDB Conv. and Maint. Aid automatically migrates logically related databases together
- HALDB does not support virtual pairing
 - ▶ Migration is to physical pairing
 - HALDB Conv. and Maint. Aid converts DBDs with virtual pairing to physical pairing
- All secondary indexes to a database must be migrated with the database
 - ▶ Indexes to HALDB databases must be HALDB (ACCESS=PSINDEX)
 - HALDB Conv. and Maint. Aid automatically migrates secondary indexes with indexed database
- HIDAM indexes are not migrated
 - ▶ HIDAM primary indexes are automatically created from PHIDAM roots



HALDB Migration Aid Utility

▲ HALDB Migration Aid utility (DFSMAID0)

- Reads existing HDAM, HIDAM, and Secondary Index databases
- Provides sizing and high key information for migration planning
 - ▶ Number of segments
 - ▶ Size of segments
 - ▶ Prefix size increases
 - ▶ Sizes of physical logical children converted from virtual logical children
- Secondary index support
 - ▶ Provides key range boundaries and numbers of records
 - ▶ Secondary index 'bytes' and 'prefix-incr' information are inaccurate in the report!
 - ▶ Number of segments and high key values are accurate in the report
 - ▶ Sizes are easily calculated from the numbers of records



HALDB Migration Aid Utility

▲ Sample report:

```
partition 1 :
```

```
minimum key =
```

```
+0000 d2c1c1f1 f1f2f3f4 |KAA11234 |
```

```
maximum key =
```

```
+0000 d2f2f3f9 f9f2f3f4 |K2399234 |
```

	segments	bytes	prefix-incr	length-incr
1) 'PRODUCT '	31567	4040576	252536	0
2) 'INVENT'	103781	8094918	830248	0
3) 'ORDQTY'	171182	10955648	1369456	0
4) 'MFGSPECS'	51115	10938610	408920	0
SUM)	357645	34029752	2861160	0

segments - number of segments

bytes - number of bytes for the segments

prefix-incr - additional bytes due to increased prefix size

length-incr - additional bytes required for paired logical relationships



HALDB Migration Aid Utility

▲ Using the Migration Aid utility

- You may specify one of the following:
 - ▶ Number of equal sized partitions
 - ▶ Number of segment bytes per partition
 - ▶ High keys for partitions
- Report for each partition and the entire database
- Bytes in reports do not include free space, bit maps, RAPs, or FSEAPs
 - ▶ You must adjust for these!

▲ HALDB Conv. and Maint. Aid has its own version of this capability



DBDGEN for HALDB

▲ Typically, minimal changes from non-HALDB

- Change HDAM to PHDAM, HIDAM to PHIDAM, or INDEX to PSINDEX on DBD statement
- Remove DATASET statements
 - ▶ Data set names are defined with Partition Definition utility
 - ▶ Data set groups are defined with DSGROUP parameter on SEGM statement
- Remove definitions of HIDAM primary index
 - ▶ These indexes are not explicitly defined with PHIDAM

▲ HALDB Conv. and Maint. Aid automatically converts DBDs

- Handles all cases including converting virtual pairing to physical pairing
- Can keep or discard data set groups

DBDGEN Example: HIDAM to PHIDAM

• HIDAM including Index

```

DBD    NAME=INDEXDB,ACCESS=INDEX
DATASET DD1=INXDDB1,
SEGM   NAME=INDEX,BYTES=21
LCHILD NAME=(SKILL,SKILLINV),INDEX=TYP1
FIELD  NAME=(INDXSEQ,SEQ,U),BYTES=21,START=1
DBDGEN
FINISH
END

```

```

DBD    NAME=SKILLINV,ACCESS=HIDAM
DATASET DD1=SKLHIDAM,BLOCK=8192,SCAN=0
SEGM   NAME=SKILL,BYTES=31,PTR=NT,PARENT=0
LCHILD NAME=(INDEX,INDEXDB),PTR=INDX
FIELD  NAME=(TYPE,SEQ,U),BYTES=21,START=1
FIELD  NAME=STDCODE,BYTES=10,START=22
SEGM   NAME=NAME,BYTES=20,PTR=T,PARENT=SKILL
FIELD  NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1
DBDGEN
FINISH
END

```

• PHIDAM

PHIDAM Index is generated automatically!
No index DBD
No LCHILD statement

No DATASET statements

Change HIDAM to PHIDAM

↓

```

DBD    NAME=SKILLINV,ACCESS=PHIDAM
SEGM   NAME=SKILL,BYTES=31,PTR=NT,PARENT=0
FIELD  NAME=(TYP1,SEQ,U),BYTES=21,START=1
FIELD  NAME=STDCODE,BYTES=10,START=22
SEGM   NAME=NAME,BYTES=20,PTR=T,PARENT=SKILL
FIELD  NAME=(STDCLEVL,SEQ,U),BYTES=20,START=1
DBDGEN
FINISH
END

```

DBDGEN Example: HDAM and PHDAM

Multiple Data Set Groups

• HDAM

```

DBD      NAME=SKILLINV ,ACCESS=HDAM ,
          RMNAME=(DFSHDC40 ,5 ,500 ,824 )
DATASET  DD1=SKILHDAM ,BLOCK=2048 ,SCAN=0

SEGM     NAME=SKILL ,BYTES=31 ,PTR=T ,PARENT=0
FIELD    NAME=(TYPE ,SEQ ,U) ,BYTES=21 ,START=1
FIELD    NAME=STDCODE ,BYTES=10 ,START=22

SEGM     NAME=NAME ,BYTES=20 ,PTR=T ,PARENT=SKILL
FIELD    NAME=(STDCLEVL ,SEQ ,U) ,BYTES=20 ,START=1

DATASET  DD1=SKILHDA2 ,BLOCK=4096 ,SCAN=0
SEGM     NAME=EXPR ,BYTES=20 ,PTR=T ,PARENT=NAME
FIELD    NAME=PREVJOB ,BYTES=10 ,START=1
FIELD    NAME=CLASSIF ,BYTES=10 ,START=11

DBDGEN
FINISH
END

```

• PHDAM

```

DBD      NAME=SKILLINV ,ACCESS=PHDAM ,
          RMNAME=(DFSHDC40 ,5 ,500 ,824 )

SEGM     NAME=SKILL ,BYTES=31 ,PTR=T ,PARENT=0
FIELD    NAME=(TYPE ,SEQ ,U) ,BYTES=21 ,START=1
FIELD    NAME=STDCODE ,BYTES=10 ,START=22

SEGM     NAME=NAME ,BYTES=20 ,PTR=T ,PARENT=SKILL
FIELD    NAME=(STDCLEVL ,SEQ ,U) ,BYTES=20 ,START=1

SEGM     NAME=EXPR ,BYTES=20 ,PTR=T ,PARENT=NAME ,
          DSGROUP=B
FIELD    NAME=PREVJOB ,BYTES=10 ,START=1
FIELD    NAME=CLASSIF ,BYTES=10 ,START=11

DBDGEN
FINISH
END

```

Change HDAM to PHDAM

DSGROUP parameter

No DATASET statements



Partition Definitions using PDU

▲ Partition Definition Utility (PDU) is ISPF based utility

- Defines partitions in a HALDB
 - ▶ Number of partitions
 - ▶ Partition selection method (key range or exit routine)
 - ▶ Partition names
 - ▶ Data set characteristics (data set names, free space, randomizing parameters, ...)
- Stores definitions in RECONs

▲ HALDB Conversion and Maintenance Aid defines partitions

- Does not require use of PDU



Sample PDU Screen (1)

Help

Partition Default Information

Command ==>

The master database information was defined successfully.
Type the field values. Then press Enter to continue.

Database Name : DAXYCUST

More: +

Processing options

Automatic definition YES

Input dataset 'ISMTESTG.CUST.KEYS(K4)'

Use defaults for DS groups . . YES

Defaults for partitions

Partition Name P7PTS%

Data Set Name Prefix IMS.DB7G.CUST

Randomizer

module name DFSHDC40

anchor 2

high block number 999

bytes 2000

Size the RAA to hold the data for the partition

F1=Help F3=Exit F6=Groups F8=Down F12=Cancel



Sample PDU Screen (2)

Help

Partition Default Information

Command ==>

Type the field values. Then press Enter to continue.

Database Name : DAXYCUST

More: - +

Free Space

free block freq. factor . . 0

free space percentage . . . 0

Defaults for data set groups

Block Size 4096

DBRC options

Max. Image Copies 2

Recovery Period 0

Recovery Utility JCL . . . RECOVJCL

Default JCL _____

Image Copy JCL ICJCL

Online Image Copy JCL . . . OICJCL

Receive JCL RECVJCL

Reusable? NO

F1=Help F3=Exit F6=Groups F7=Up F8=Down F12=Cancel



Partition Definitions using DBRC Commands

▲ DBRC Commands may be used to define partitions

- An alternative to the use of the Partition Definition Utility
- INIT.DB specifies the database
- INIT.PART is used to define a partition

```
INIT.DB    DBD(DAXYCUST) TYPHALDB SHARELVL(3)
```

```
INIT.PART DBD(DAXYCUST)  PART(CUSTC1) KEYSTRNG(0006666600) -  
          DSNPREFIX(IMS.DB7G.CUST)    -  
          GENMAX(3)
```

```
INIT.PART DBD(DAXYCUST)  PART(CUSTC2) KEYSTRNG(0013333800) -  
          DSNPREFIX(IMS.DB7G.CUST)    -  
          GENMAX(3)
```

```
INIT.PART DBD(DAXYCUST)  PART(CUSTC3) KEYSTRNG(0020000000) -  
          DSNPREFIX(IMS.DB7G.CUST)    -  
          GENMAX(3)
```



Database Data Set Allocation

▲ User must allocate database data sets

- Data set names specified in Partition Definition Utility and stored in RECONs

▲ DBRC GENJCL.USER may be used to create allocation jobs

- Create 4 skeletal JCL members with user variables
 - IDCAMS DEFINEs for ILDSs, INDEXes, and ESDSs
 - IDCAMS ALLOCATE for OSAM data sets
 - ▶ Use these 4 members for all databases
- Create DBDSGRPs for each database
 - ▶ One group for each type of data set in the database
 - ILDSs, INDEX, and OSAM or VSAM
- "Using GENJCL.USER to Allocate IMS HALDB Data Sets" at www.ibm.com/support/techdocs/atmastr.nsf/PubAllNum/TD100491 explains this technique and provides example code

▲ IMS HALDB Conv. and Maint. Aid allocates these data sets for you



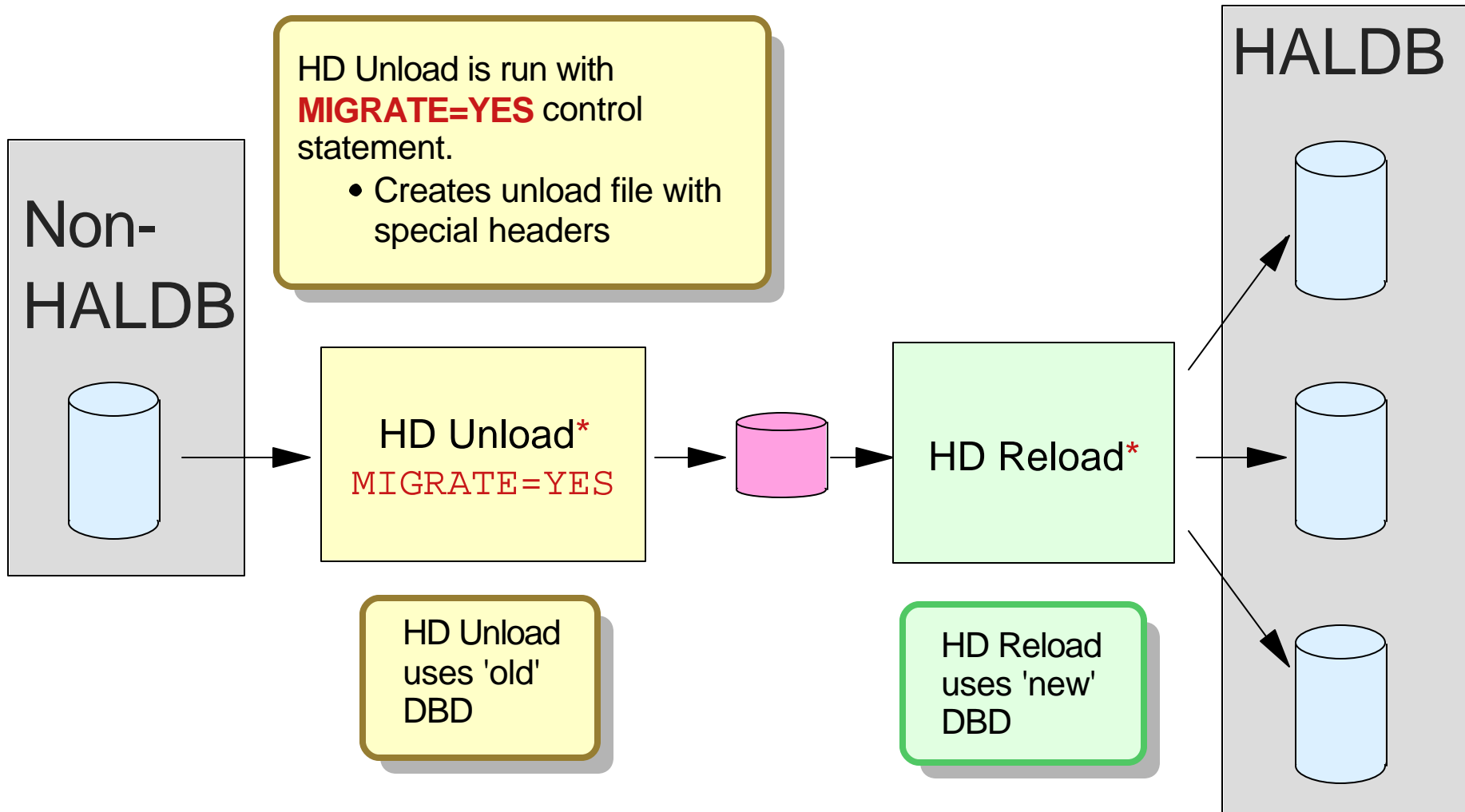
Database Initialization

- ▲ **Initialization is a new function of the Prereorganization utility ***
 - Sets "high used RBA" to non-zero value
 - Makes partition usable
 - ▶ Partitions with no data are valid
 - Utility is run for database
 - ▶ "Partition initialization required" state is kept in RECONS for each partition
 - ▶ Only partitions requiring initialization are initialized

- ▲ **HALDB Conversion and Maintenance Aid invokes this function**

- * IBM High Performance Load may be used in place of Prereorganization utility

Migrating to HALDB



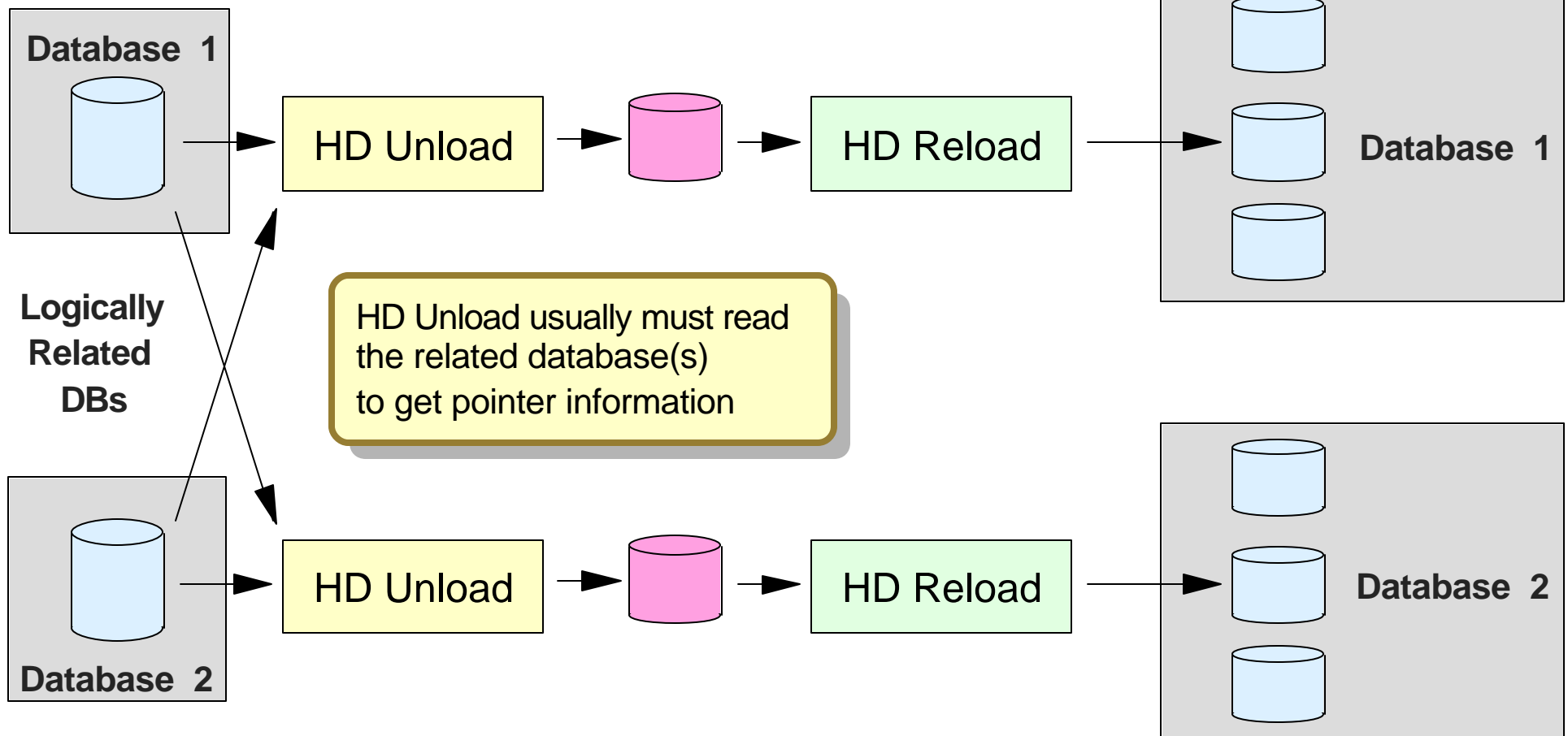
* HALDB Conv. and Maint. Aid creates the proper Unload and Reload jobs

* IBM HP Unload and HP Load may be used in place of HD Unload and HD Load

Migrating DBs with Logical Relationships

Non-HALDB

HALDB





Migrating DBs with Logical Relationships

▲ Logically related databases are read by HD Unload when:

- ▶ Using physical pairing
 - ▶ All uses of VIRTUAL option (LPCK not stored in log. child segment)
 - ▶ All uses of symbolic pointers
 - ▶ When unloading virtual logical children
-
- DFSVSAMP must provide buffer pools for these databases

 - Reads of logically related DBs are likely to be random
 - ▶ Could be many I/Os

 - Concurrent unloads of logically related databases may affect performance of each other

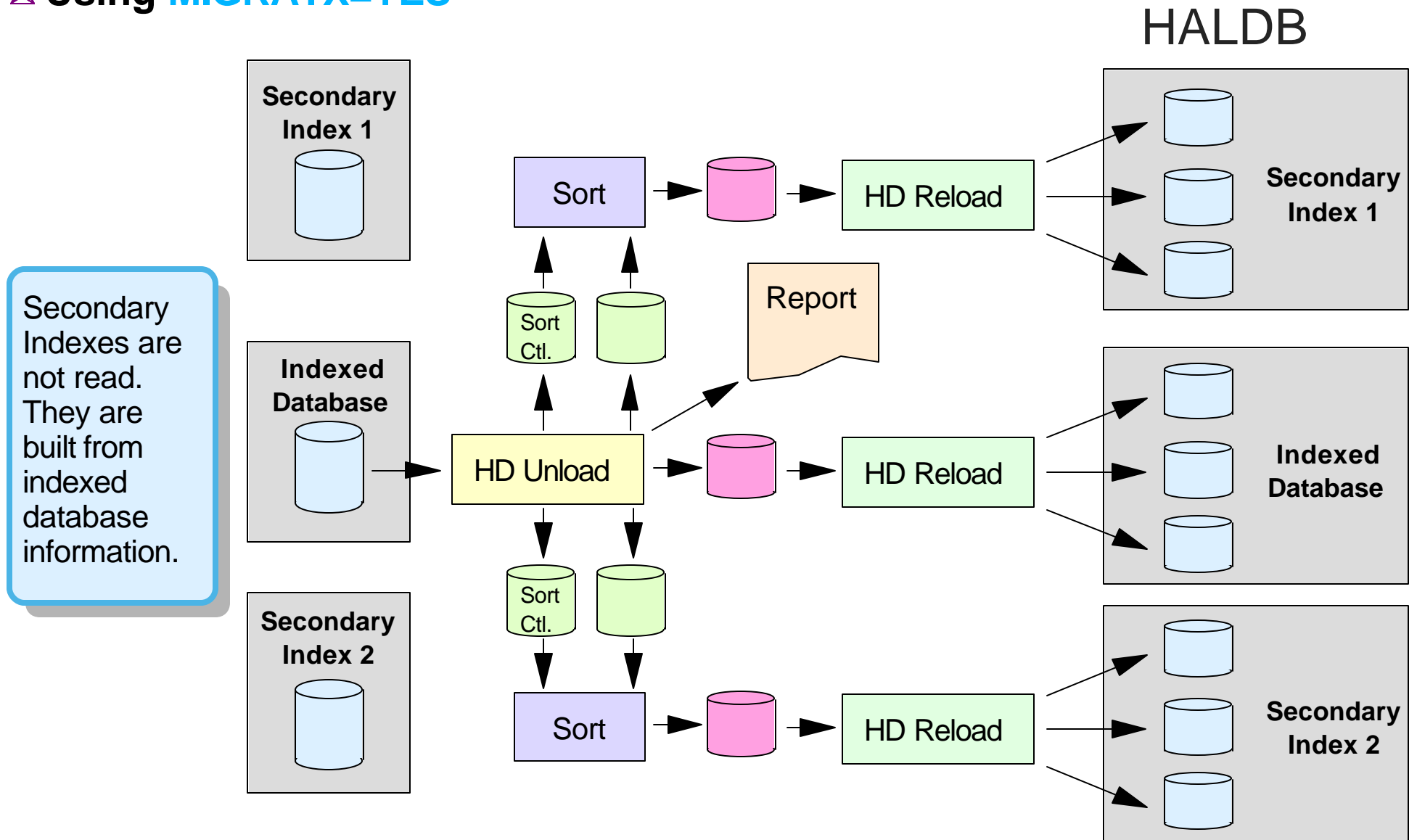


Migrating Secondary Indexes

- ▲ **HD Unload indexed database with `MIGRATX=YES` control statement**
 - ▶ Creates unload file for each secondary index
 - Sort unload files
 - HD Reload indexed DB and each secondary index

Migrating Secondary Indexes

▲ Using MIGRATX=YES



MIGRATX=YES with Secondary Indexes

▲ Sample report:

W O R K F I L E S T A T I S T I C S					
SINAME	WFNAME	SFNAME	RCDTOTAL	OFFSET	LENGTH
ABCSI01	DFSWRK01	DFSSRT01	00000075	0069	0018
DEFSI02	DFSWRK02	DFSSRT02	00000150	0069	0018

Diagram annotations: Arrows point from the label "DDNAMEs" to the WFNAME and SFNAME columns. An arrow points from the label "Sort Field" to the OFFSET and LENGTH columns.

▲ Secondary index is not read by HD Unload

- User data is not migrated

▲ Indexed database is read once by HD Unload for all secondary index migrations:

- DFSVSAMP must provide buffer pools for this database



Creating the ILDS During Migration

▲ HD Reload has three options for ILDS creation:

1. Update ILDSs randomly as target segments are loaded
 - ▶ Part of HD Reload execution
2. Sort ILDS update records and apply updates sequentially
 - ▶ Part of HD Reload execution
3. Do not update ILDSs during Reload
 - ▶ Create ILDSs with Index/ILDS Rebuild utility



Creating the ILDS - Migration Option 1

▲ HD Reload without a control statement

- Updates ILDSs as target segments are loaded
 - ▶ ILDS writes done randomly in update mode
- Single thread process for multiple ILDSs
 - ▶ Each partition has its own ILDS
- Can be time consuming



Creating the ILDS - Migration Option 2

▲ HD Reload with ILDSMULTI control statement

- Multithread process with thread for each ILDS (partition)
 - ▶ Sorts entries using data spaces
- Sequentially writes entries in load mode
 - ▶ No CI/CA splits
 - ▶ Honors KSDS free space specifications
 - ▶ Helps next reorganization
- Provides better performance



Creating the ILDS - Migration Option 3

▲ HD Reload with NOILDS control statement

- HD Reload does not write ILDS entries
- Marks all ILDSs as 'Recovery Needed' in RECONS
- ILDSs must be created with Index/ILDS Rebuild utility
 - ▶ Utility executions for ILDSs are independent
 - ▶ May be run in parallel or serially
 - ▶ May be run on same or different processors
- Spreads CPU, storage, and DASD use across multiple jobs



Migrating from Previous Partitioning

▲ Partitioning Products for IMS databases:

- IBM - IMS/ESA Partition Support Product (**PDB**) (5697-D85)
- Neon - Partitioned Database Facility (**PDF**)

▲ User partitioning

- Multiple IMS databases viewed as one database
 - ▶ Application selects partition
 - or
 - ▶ Language interface module modified to select partition



Migrating from PDB

▲ PDB characteristics:

▲ HIDAM, HISAM, and Secondary Index support

- Partitions by key range
- HISAM must be converted to PHIDAM to use HALDB

▲ HDAM support

- Partition selection by HDAM randomizer
 - ▶ RAP range per partition
or
 - ▶ Two-stage randomizer



Migrating from PDB

▲ Maintaining the same partition boundaries

- HIDAM and Secondary Indexes
 - ▶ HALDB partitions must be by key range
- HDAM
 - ▶ HALDB Partition Selection exit must be used to match PDB/PDF RAP ranges

▲ Changing partition boundaries

- Any scheme acceptable to HALDB may be used

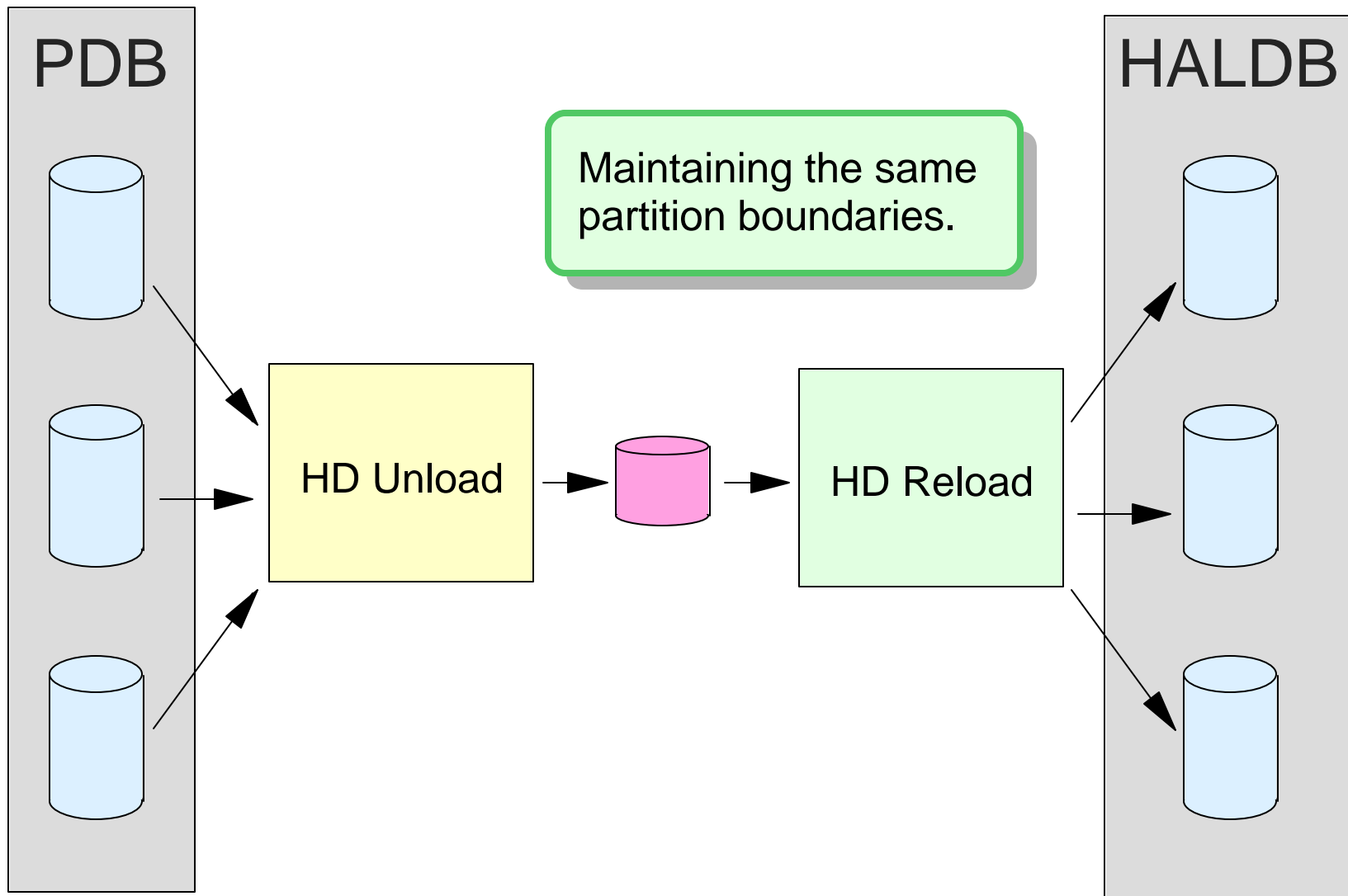


Migration from PDB with HD Unload/Reload

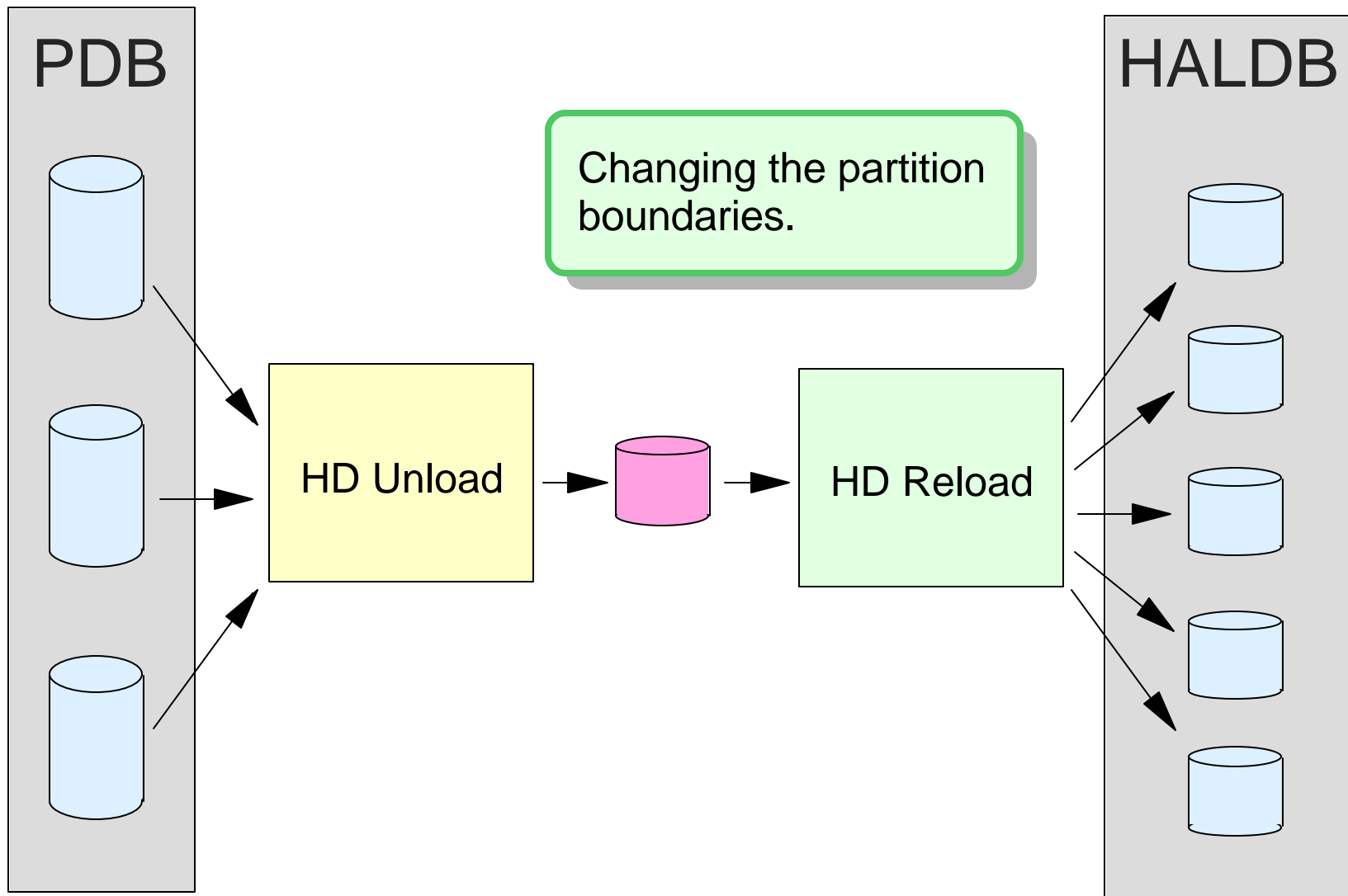
▲ Using HD Unload and Reload

- HD Reload requires HD Unload output produced with MIGRATE=YES or MIGRATX=YES control statement
 - ▶ MIGRATX=YES required with PDB/PDF secondary indexes
- HD Unload processes entire database
 - ▶ Not just a partition
- Therefore,
 - ▶ Entire database must be migrated
 - ▶ Migration of partitions in parallel cannot be done
- HALDB Conv. and Maint. Aid will invoke HD Unload and Reload for the migration

PDB to HALDB - Example 1



PDB to HALDB - Example 2



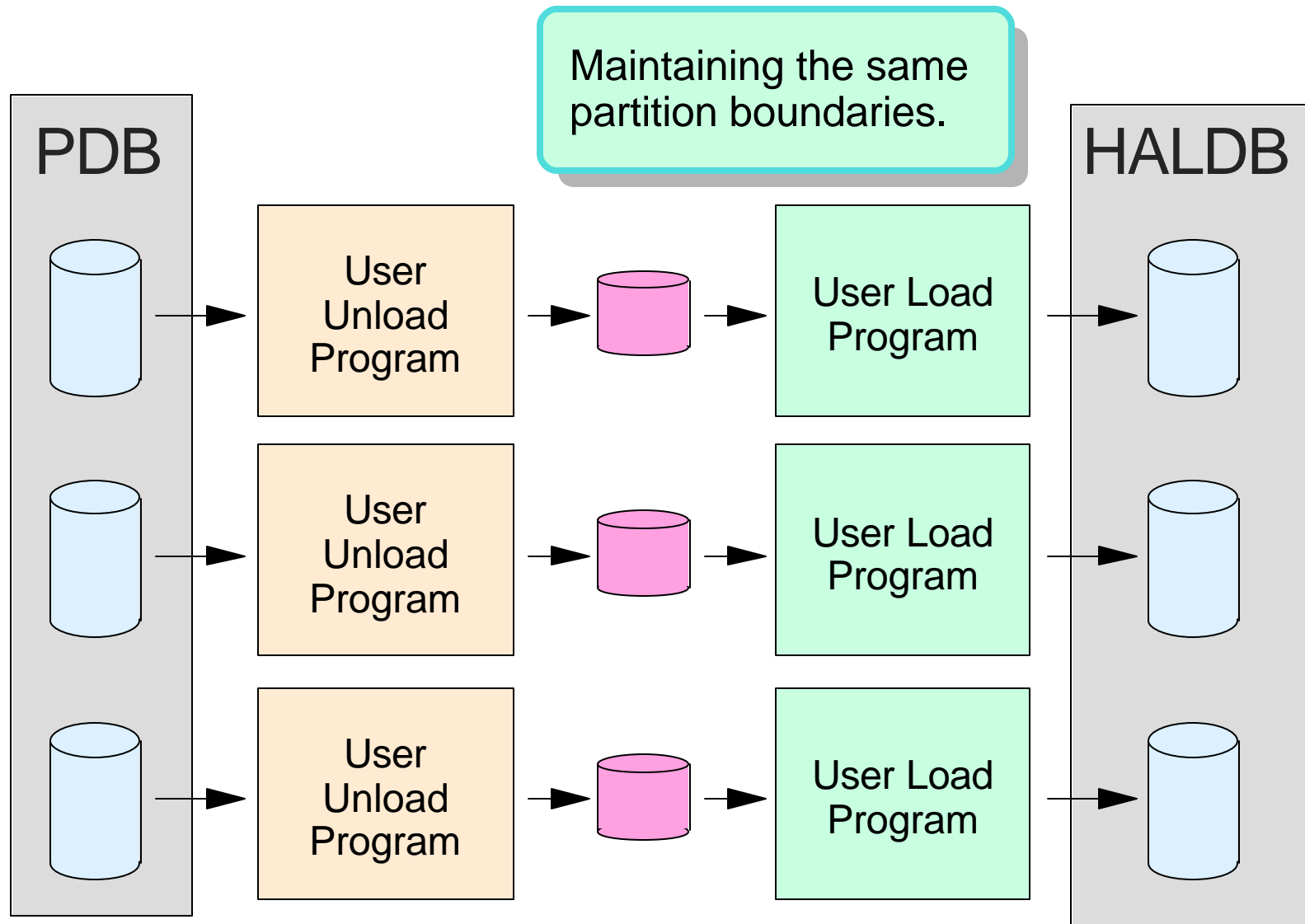


Migration from PDB with Appl. Programs

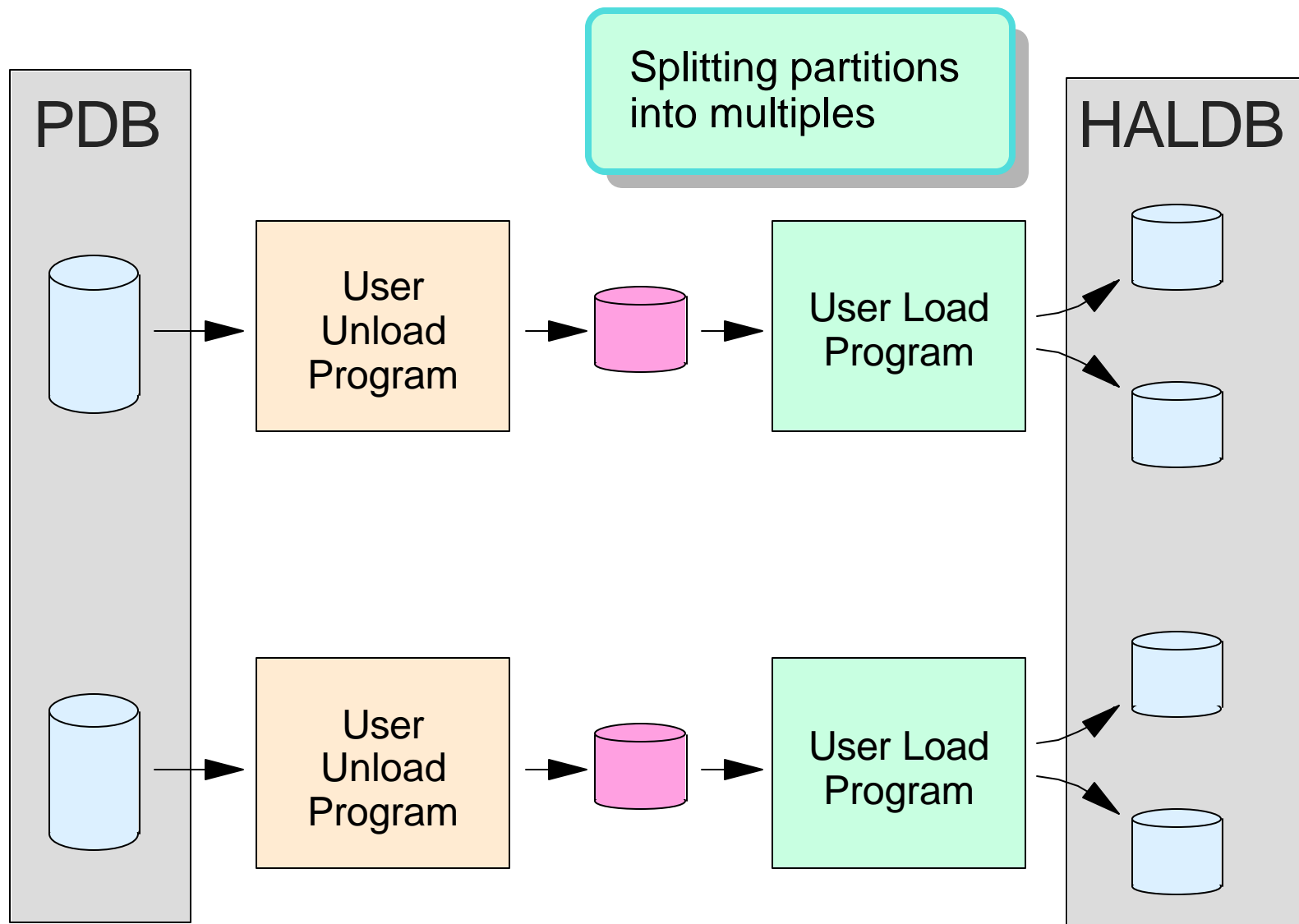
▲ Using application programs

- Read partitions with parallel executions
 - ▶ Create files to be read by load programs
- Load partitions with parallel executions
 - ▶ Without logical children, PROCOPT=L may be used
 - ▶ If logical children exist,
Load logical parents, then insert logical children using PROCOPT=A

PDB to HALDB - Example 3



PDB to HALDB - Example 4





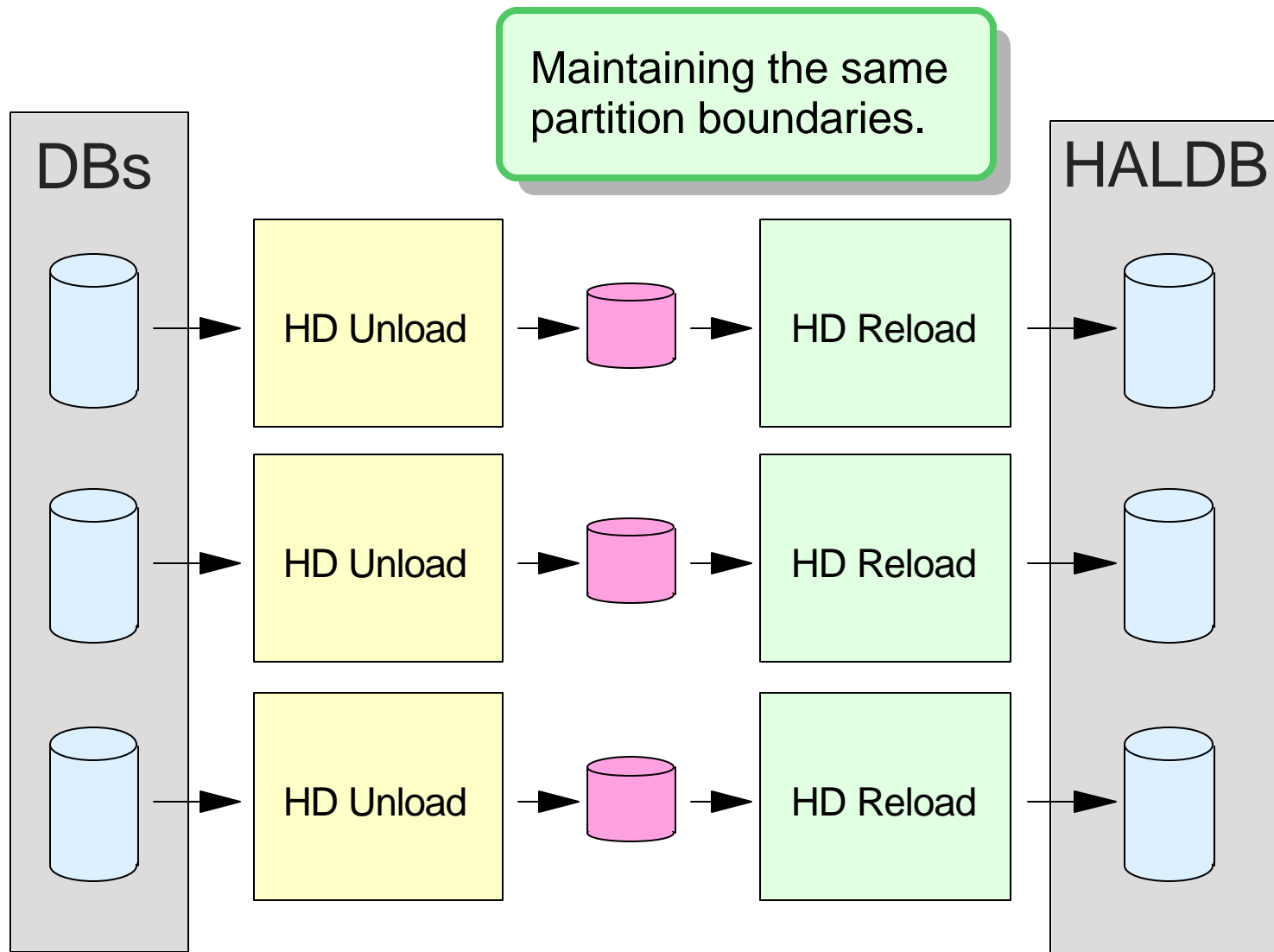
Migrating from User Partitioning

▲ Partition boundaries are determined by implementation

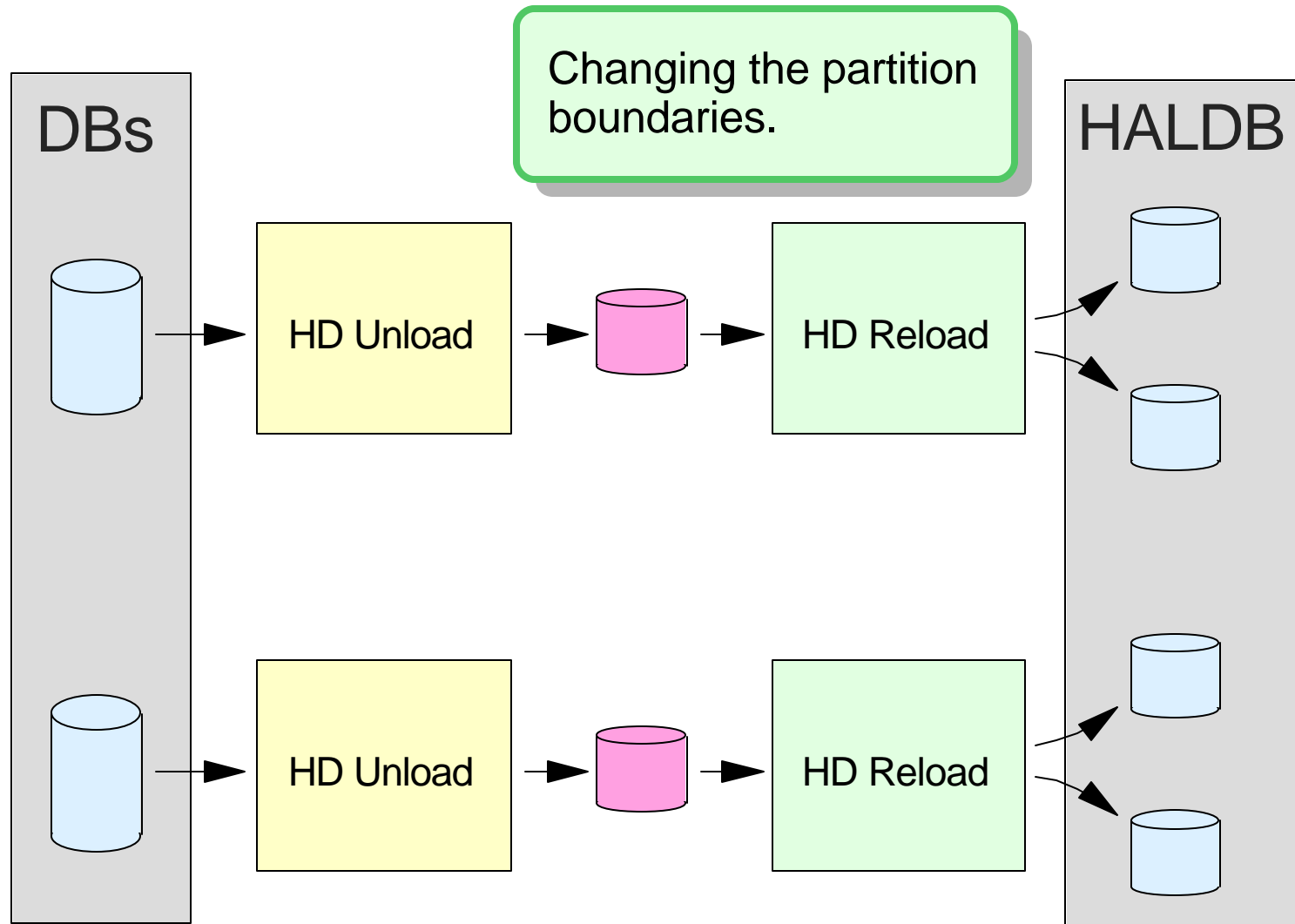
▲ Migrating partitions

- Each user partition requires execution of Unload and Reload
- If partition boundaries remain unchanged
 - ▶ Each physical database becomes a HALDB partition
 - ▶ HD Unload for each user partitioned database
 - ▶ HD Reload for each HALDB partition

User Partitioning to HALDB - Example 1



User Partitioning to HALDB - Example 2

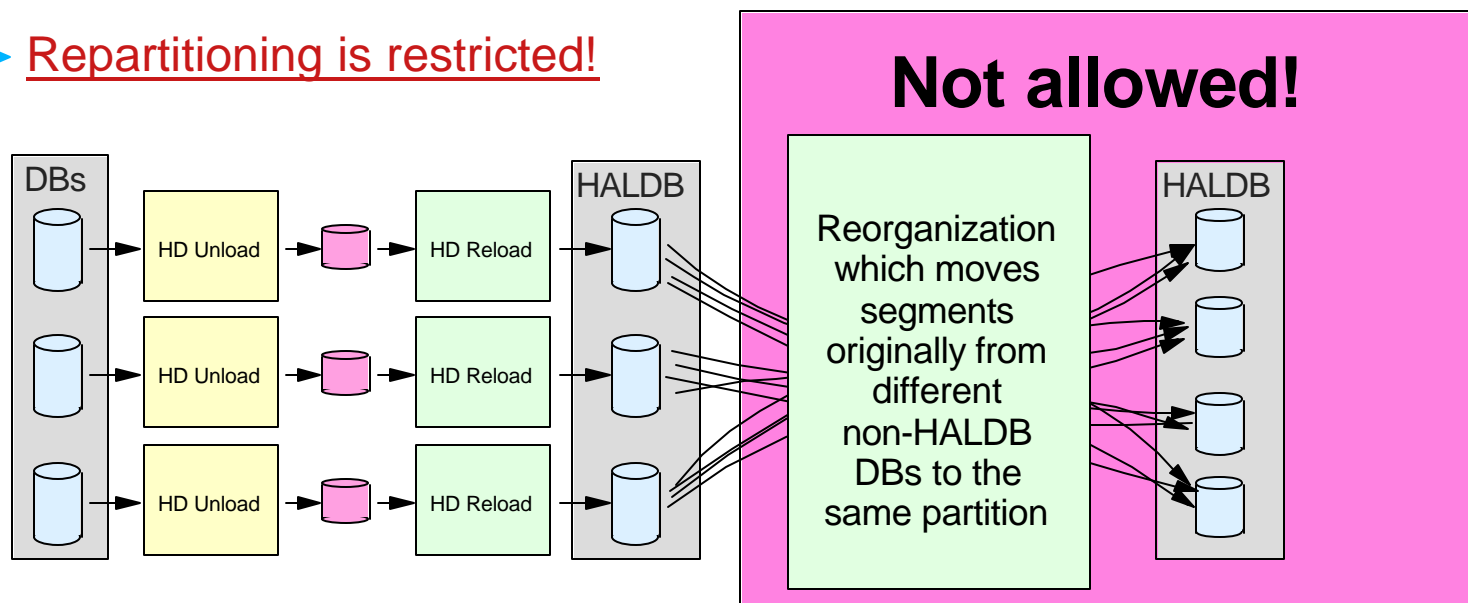


Warning on Repartitioning

▲ Restriction on using multiple executions of HD Reload with logical relationships or secondary indexes:

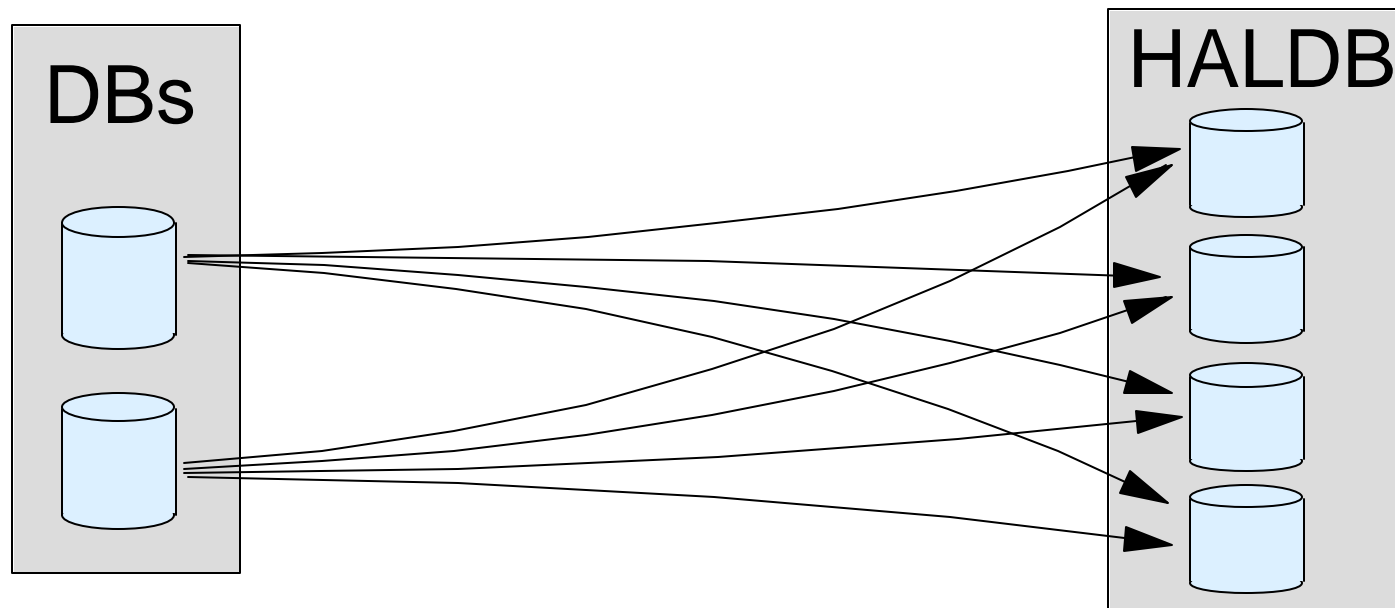
- Target segments created by different executions of HD Reload can never be moved to the same partition
 - ▶ ILK of segment is based on location (RBA) in non-HALDB database
 - Duplicates could exist from different databases

▶ Repartitioning is restricted!



Migrating from User Partitioning

- ▲ **Most migrations require initial load of HALDB partitions, possibly followed by updates of HALDB partitions**
 - Allow repartitioning without restrictions
 - Allow changes of partition boundaries from user partitioning
 - ▶ Records from multiple user partitions moved to one HALDB partition





▲ Migration Steps

- DBDGEN for HALDB
- Define partitions with Partition Definition Utility
- Allocate database data sets
- Migrate logically related DBs and secondary indexes together
- Non-HALDB databases are unloaded with HD Unload
 - ▶ MIGRATE=YES or MIGRATX=YES required
- HALDB databases are created with HD Reload
 - ▶ Steps may be affected by MIGRATE or MIGRATX choice
 - ▶ Options for creating ILDSs (applicable for secondary indexes and logical relationships)

▲ HALDB Conversion and Maintenance Aid simplifies the process

- Invokes or replaces each step