S45

Migrating to HALDB

Rich Lewis

IMS Technical Conference

Miami Beach, FL October 22-25, 2001

RETURN TO INDEX

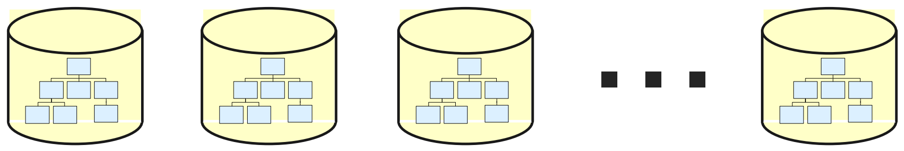
Abstract and Trademarks

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:

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 <u>does not require</u> changes to secondary indexes or logically related databases which point to it

HALDB Overview - Pointers

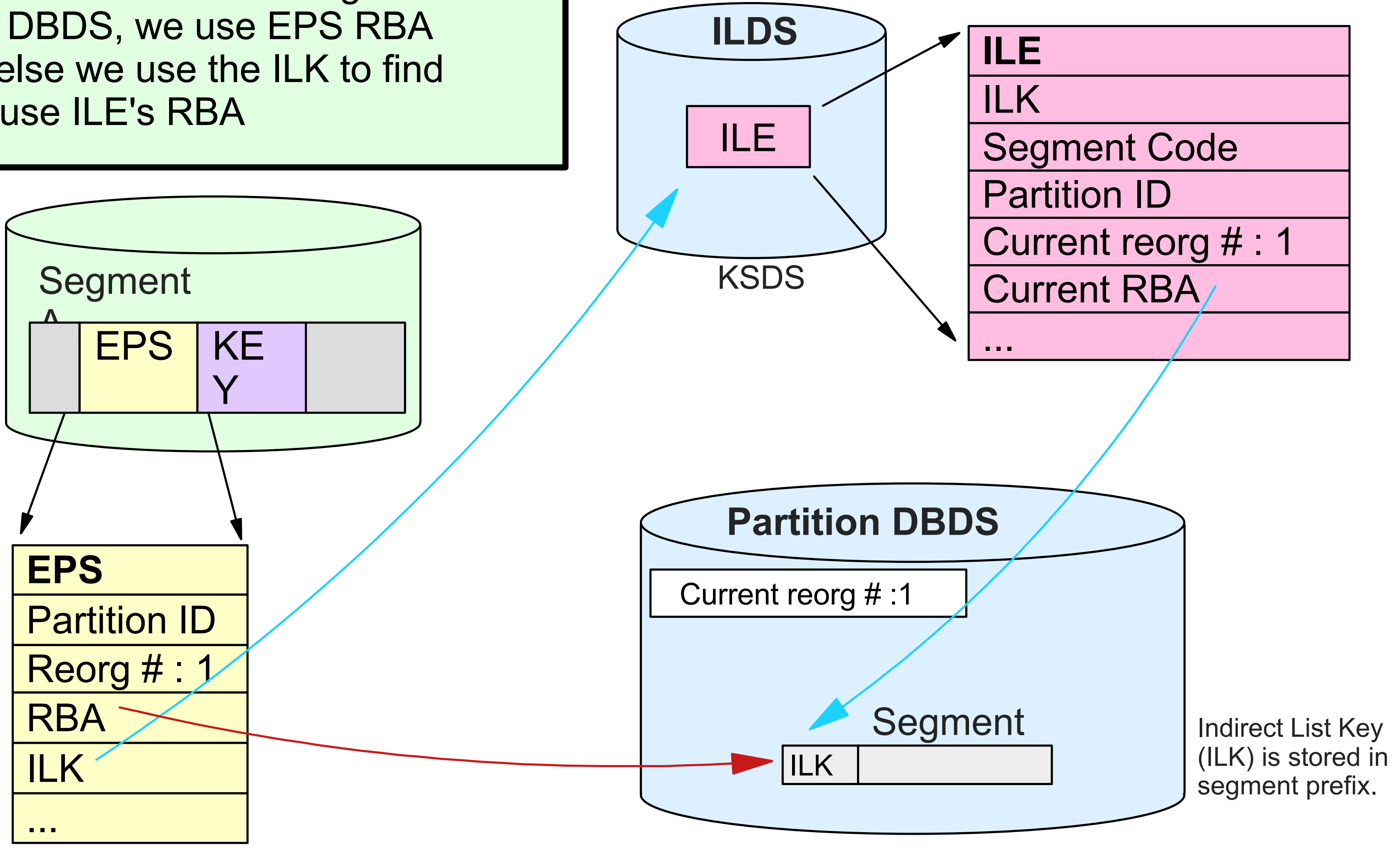
- **Logical child segments and secondary index segments include:**
  - Key of target
    - Key of target's root for secondary indexes
    - Logical Parent's concatenated key of for logical relationships
    - Used to determine partition in which target resides
  - Extended Pointer Set (EPS)
    - Reorganization 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 database
  - HALDB does not support virtual pairing
    - Migration is to physical pairing
  - All secondary indexes to a database must be migrated with the database
    - Indexes to HALDB databases must be HALDB (ACCESS=PSINDEX)
  - HIDAM indexes are not migrated
    - HIDAM primary indexes are automatically created from PHIDAM roots

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

DBDGEN Example: HIDAM to PHIDAM

- **HIDAM including Index**

```
DBD    NAME=INDEXDB,ACCESS=INDEX
DATASET DD1=INDXDB1,
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

Change HDAM to PHDAM

- **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
```

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

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  . . . . . . . : SRREGION
                                                                    More:    +
                          Processing options
  Automatic definition . . . . . YES
  Input dataset  . . . . . . . . 'imstestg.parts.keys(k4)'
  Use defaults for DS groups . . YES

                        Defaults for partitions
  Partition Name . . . . . . . . regn%%%
  Data Set Name Prefix . . . . . IMS.SRREGION.YEAR1998

  Randomizer
     module name . . . . . . . . DD41DUP2
     anchor  . . . . . . . . . . 2
     high block number . . . . . 999
     bytes . . . . . . . . . . . 2000

   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  . . . . . . . . : SRREGION
                                                                    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
```

Database Data Set Allocation

- **User must allocate database data sets**
  - Data set names specified in Partition Definition Utility and stored in RECONs

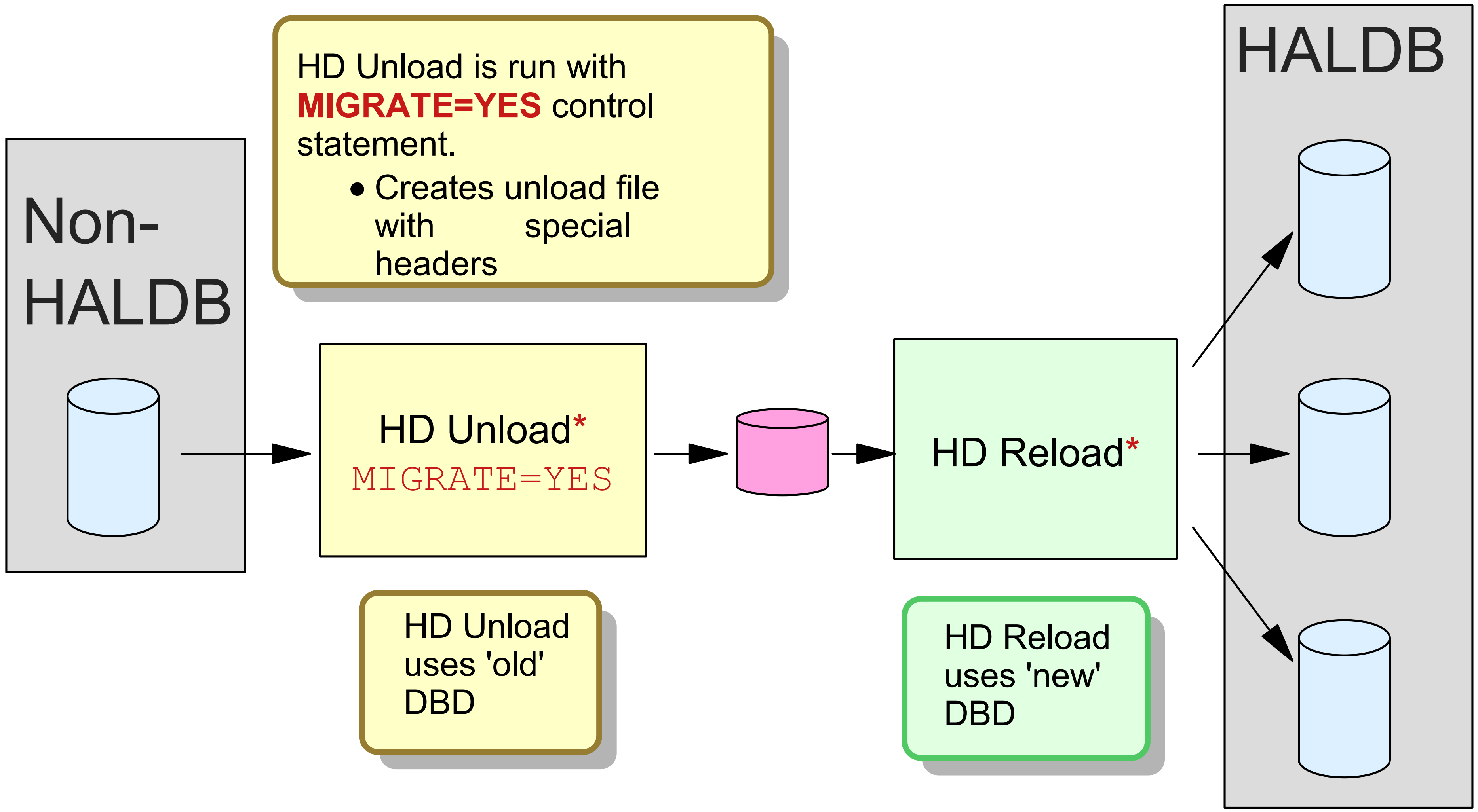
- **DBRC GENJCL.USER may be used to create allocation jobs**
  - Create DBDSGRPs
    - Group for all ILDSs in a database
    - Group for all INDEX data sets in a database
    - Group for all ESDS or OSAM data sets in a database
  - Create 4 skeletal JCL members
      - IDCAMS DEFINE for ILDSs
      - IDCAMS DEFINE for INDEXes
      - IDCAMS DEFINE for ESDSs
      - IEFBR14 allocation for OSAM data sets
    - Include user variables for primary and secondary allocation values
    - Use these 4 members for all databases

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
    - Only partitions requiring initialization are initialized

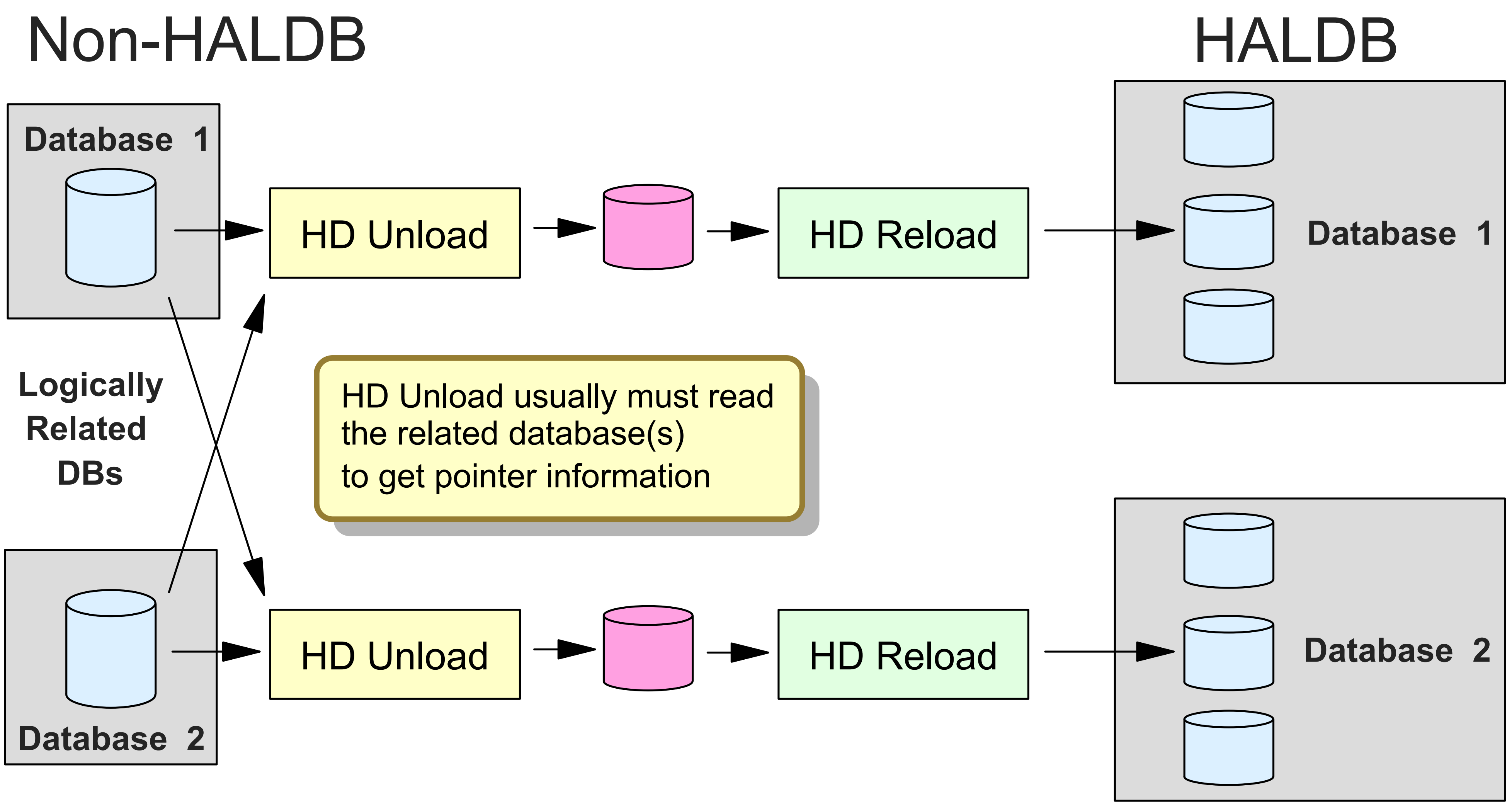
* IBM High Performance Load may be used in place of Prereorganization utility

Migrating to HALDB


HD Unload is run with **MIGRATE=YES** control statement.
• Creates unload file with special headers
Non-HALDB
HD Unload*
MIGRATE=YES
HD Reload*
HALDB
HD Unload uses 'old' DBD
HD Reload uses 'new' DBD

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

Migrating DBs with Logical Relationships



HD Unload usually must read the related database(s) to get pointer information

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

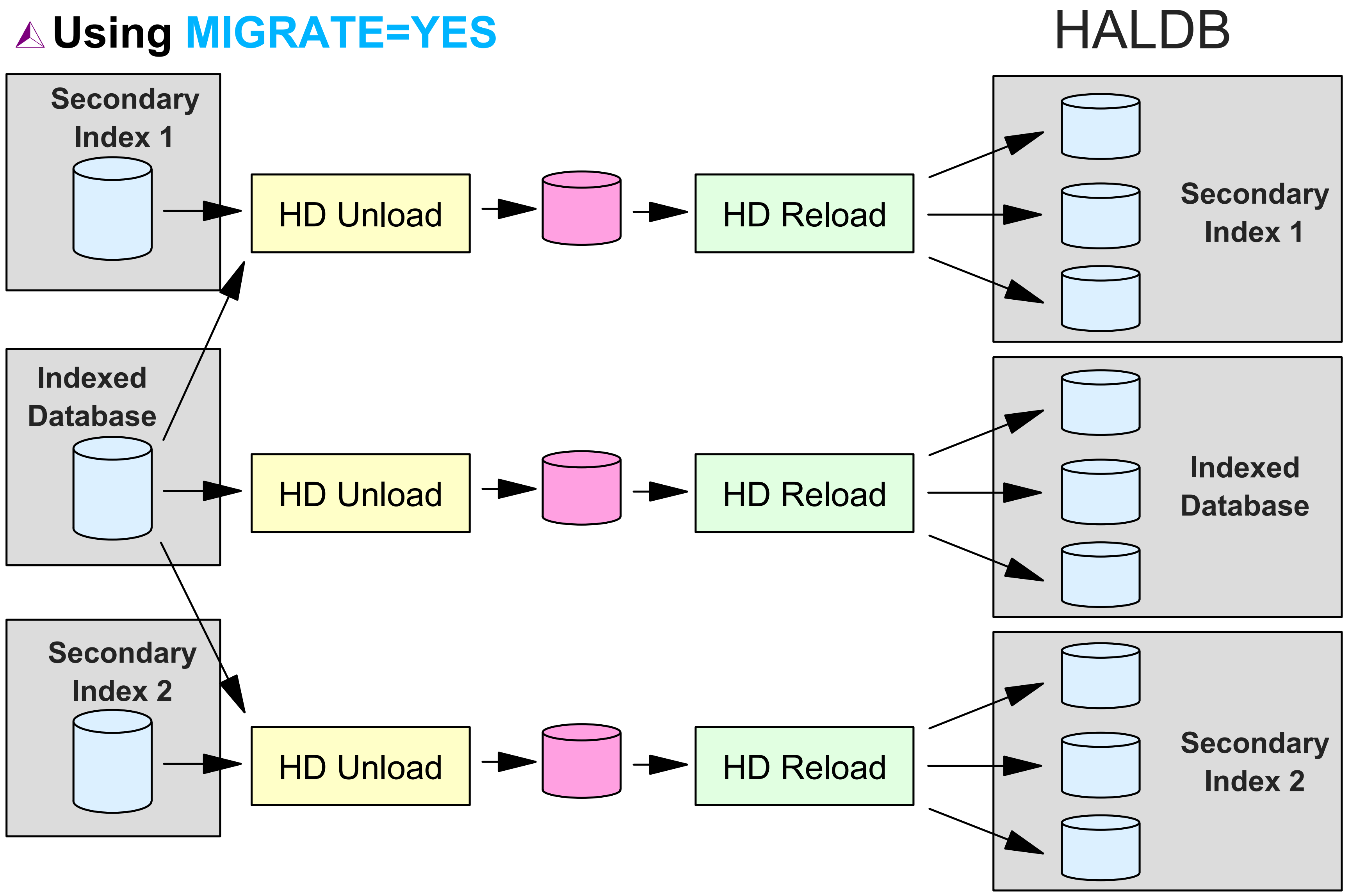
- Concurrent unloads of logically related databases may affect performance of each other

Migrating Secondary Indexes

- **Two methods for migrating secondary indexes**
  - Method 1:
    - HD Unload each secondary index with **MIGRATE=YES** control statement
    - HD Reload each secondary index
  - Method 2:
    - HD Unload indexed database with **MIGRATX=YES** control statement
      - Creates unload file for each secondary index
    - Sort unload files
    - HD Reload each secondary index

Migrating Secondary Indexes

- Using MIGRATE=YES



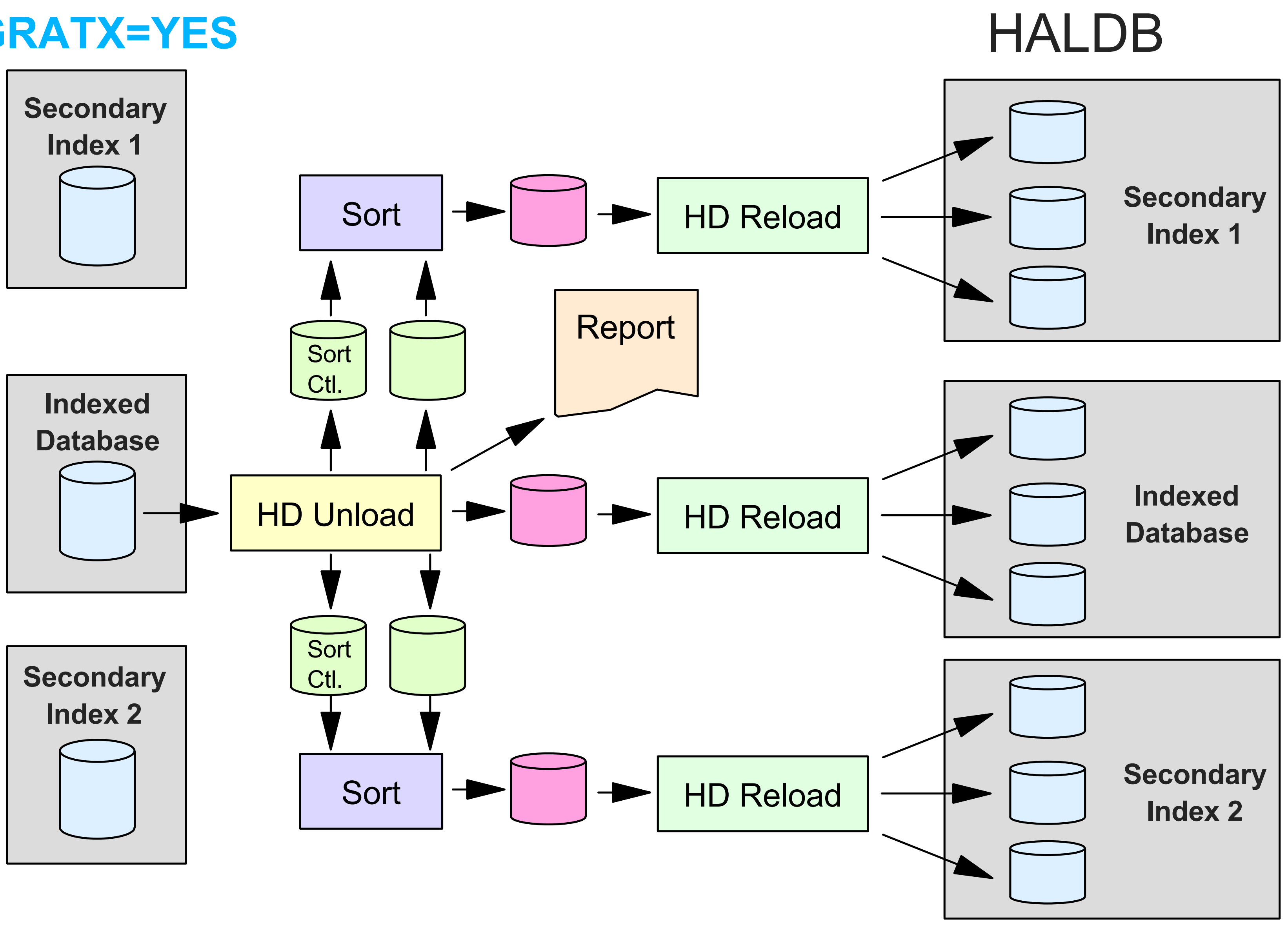
MIGRATE=YES with Secondary Indexes

- **Secondary index is read by HD Unload**
  - DFSVSAMP must provide buffer pools for this database
- **Indexed database is read by HD Unload**
  - DFSVSAMP must provide buffer pools for this database
  - Reads of indexed database are likely to be random
    - Could be many I/Os
  - Concurrent unloads of secondary indexes pointing to the same database may affect performance of each other
  - User data is maintained

Migrating Secondary Indexes

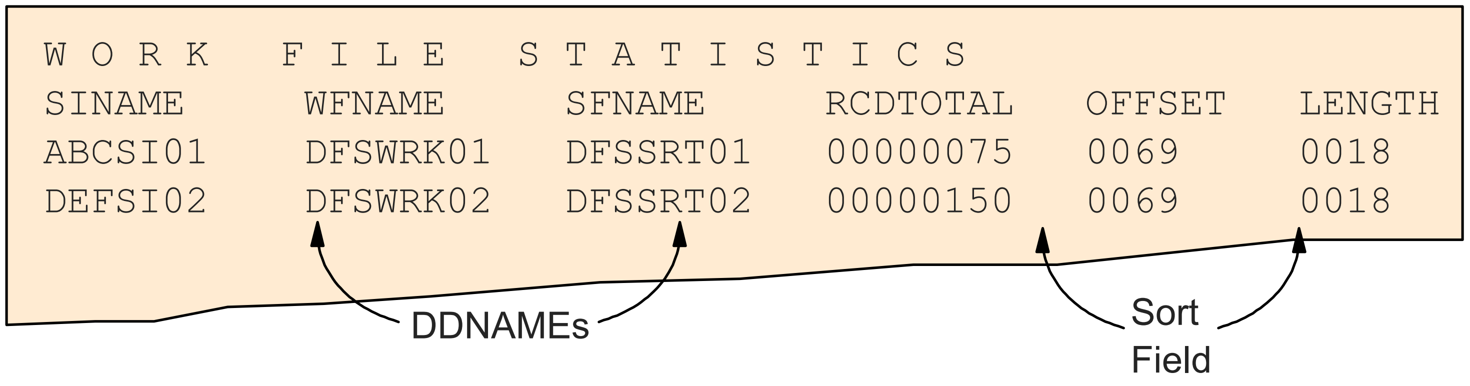
- **Using MIGRATX=YES**

Secondary Indexes are not read. They are built from indexed database information.



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
```

DDNAMEs
Sort Field

- **Secondary index is <u>not</u> read by HD Unload**
  - User data is not migrated

- **Indexed database is read <u>once</u> by HD Unload for all secondary index migrations:**
  - DFSVSAMP must provide buffer pools for this database

- **This method tends to be much more efficient**

Creating the ILDS During Migration

- **HD Reload has three options for ILDS creation:**

  1. Update ILDSs randomly as target segments are loaded
  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

- Mulithread 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

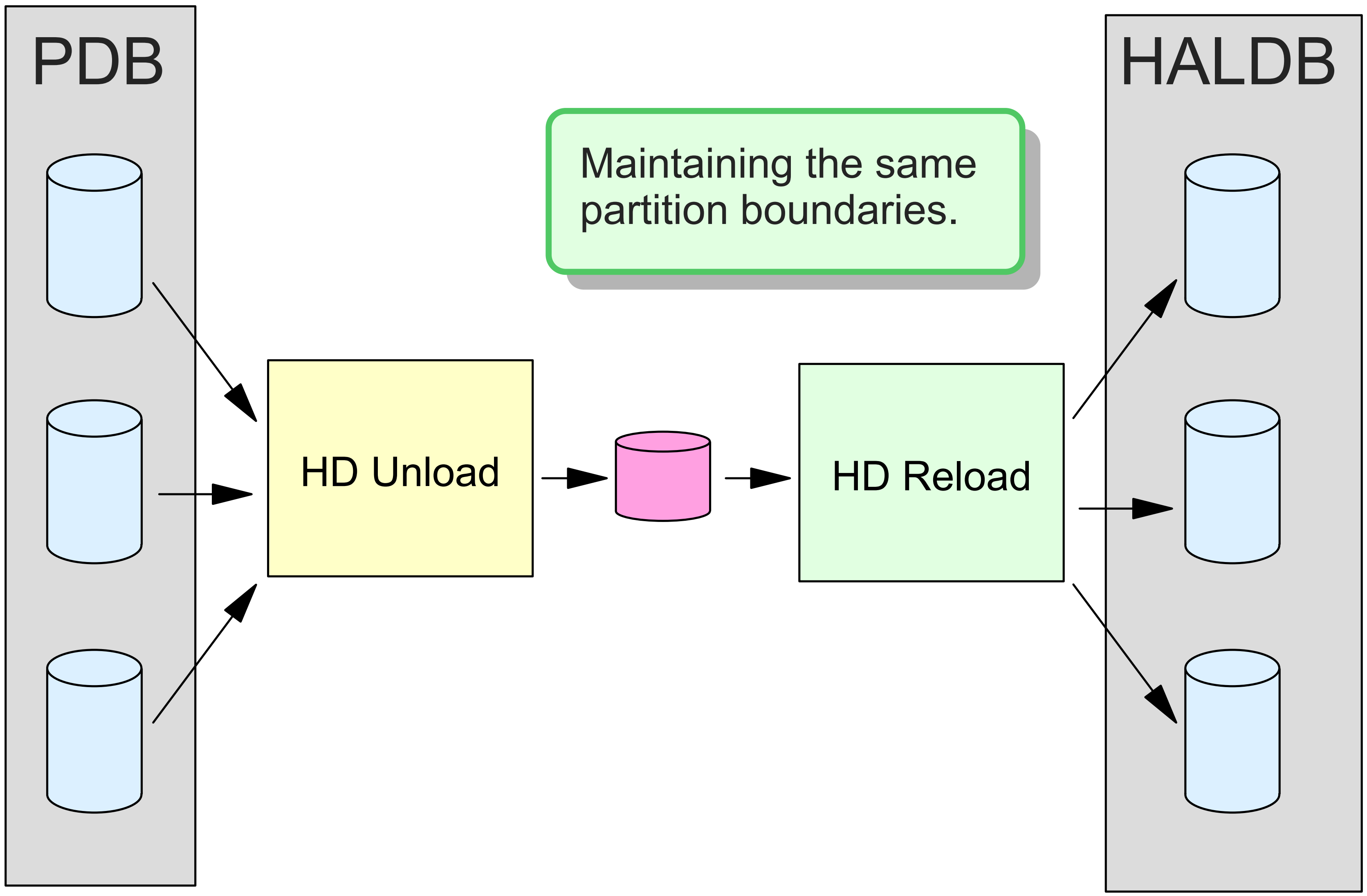
- **Maintaining the same partition boundaries**
  - HIDAM and Secondary Indexes
    - HALDB partitions must be by key range
  - HDAM
    - HALDB Partition Selection exit must be used to match PDB/PDF RAP ranges
- **Changing partition boundaries**
  - Any scheme acceptable to HALDB may be used

Migration from PDB with HD Unload/Reload

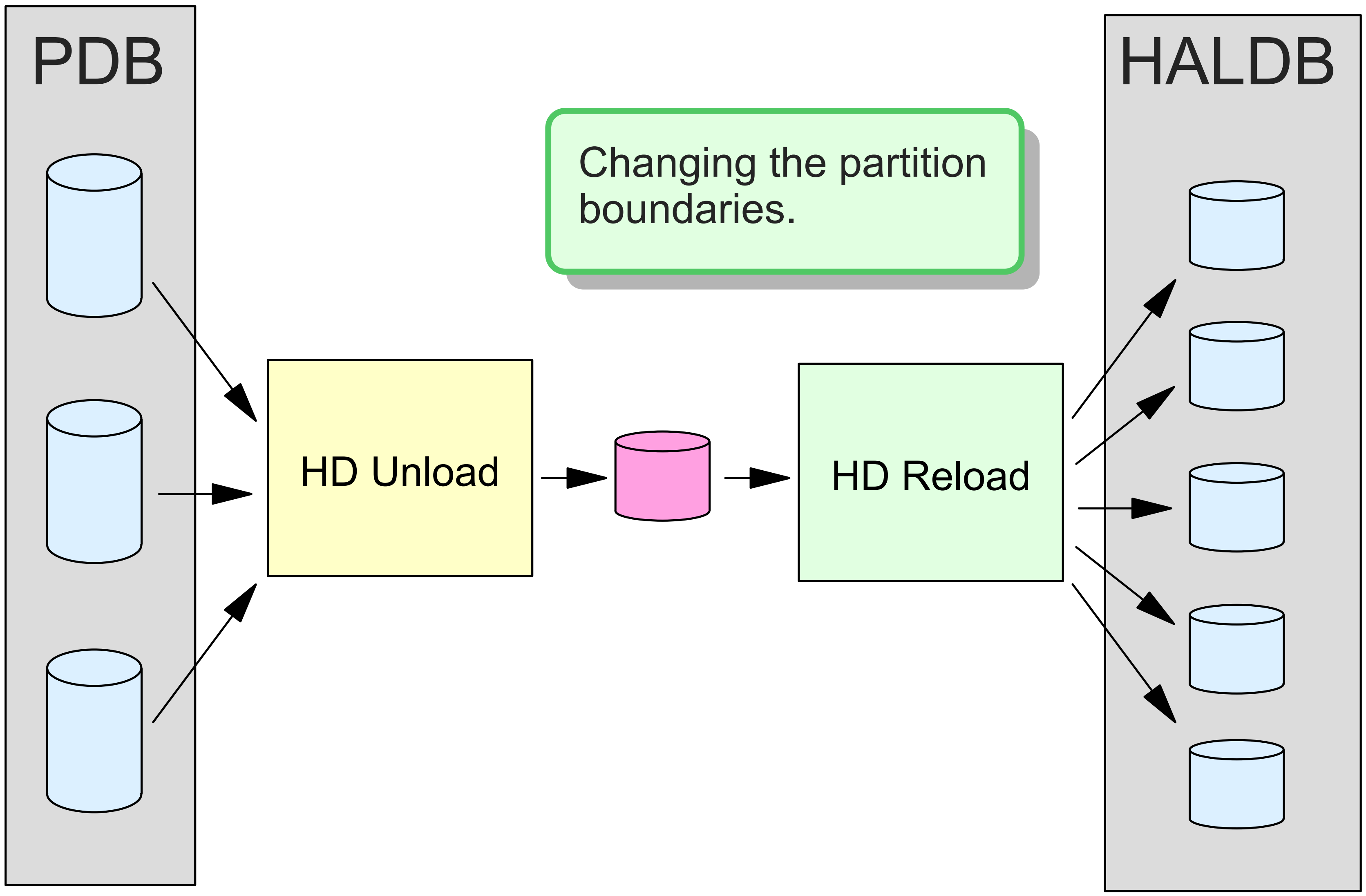
Using HD Unload and Reload

- 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

PDB to HALDB - Example 1



PDB to HALDB - Example 2



Changing the partition boundaries.

Migration from PDB with Appl. Programs

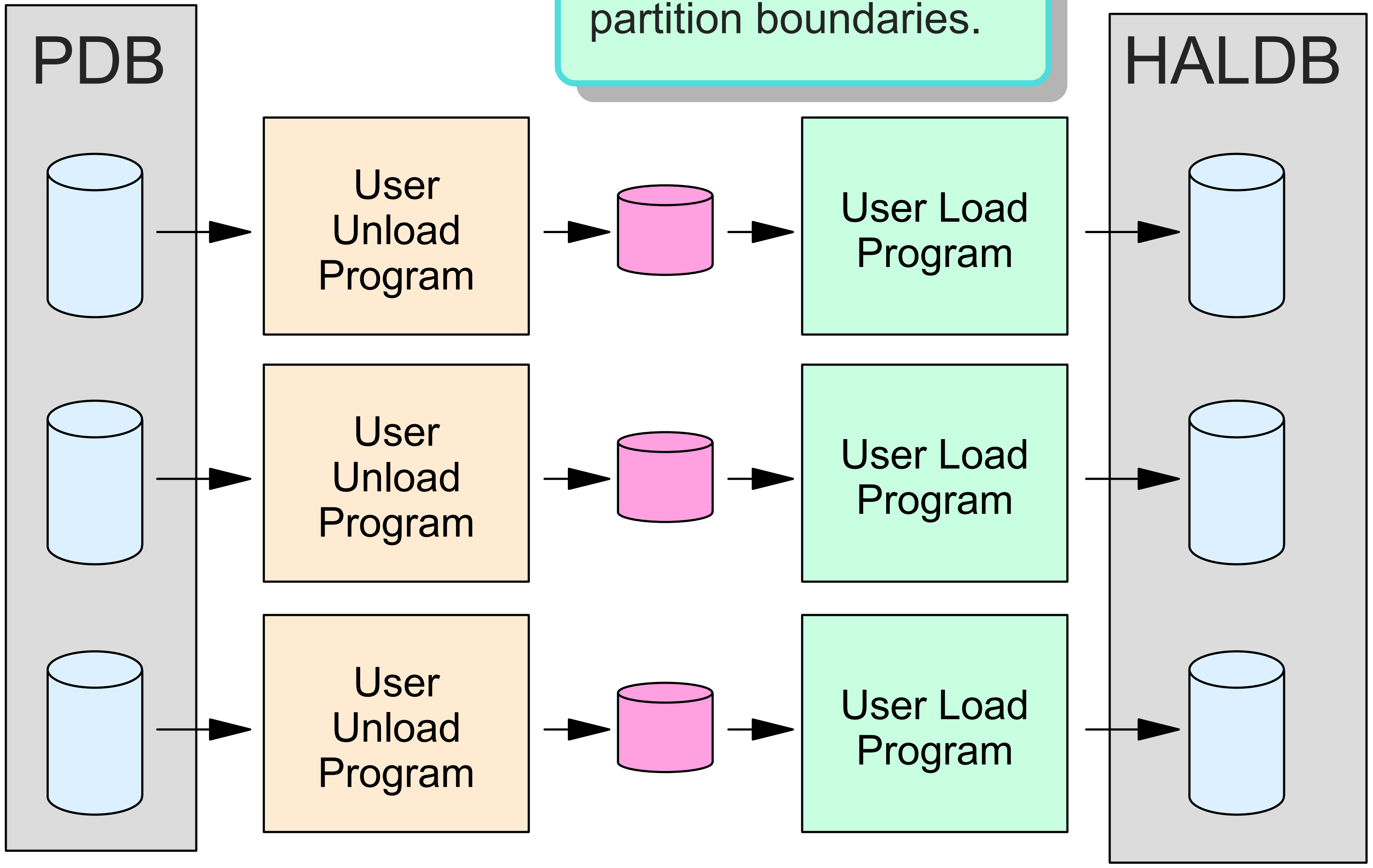
Using application programs

- 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,

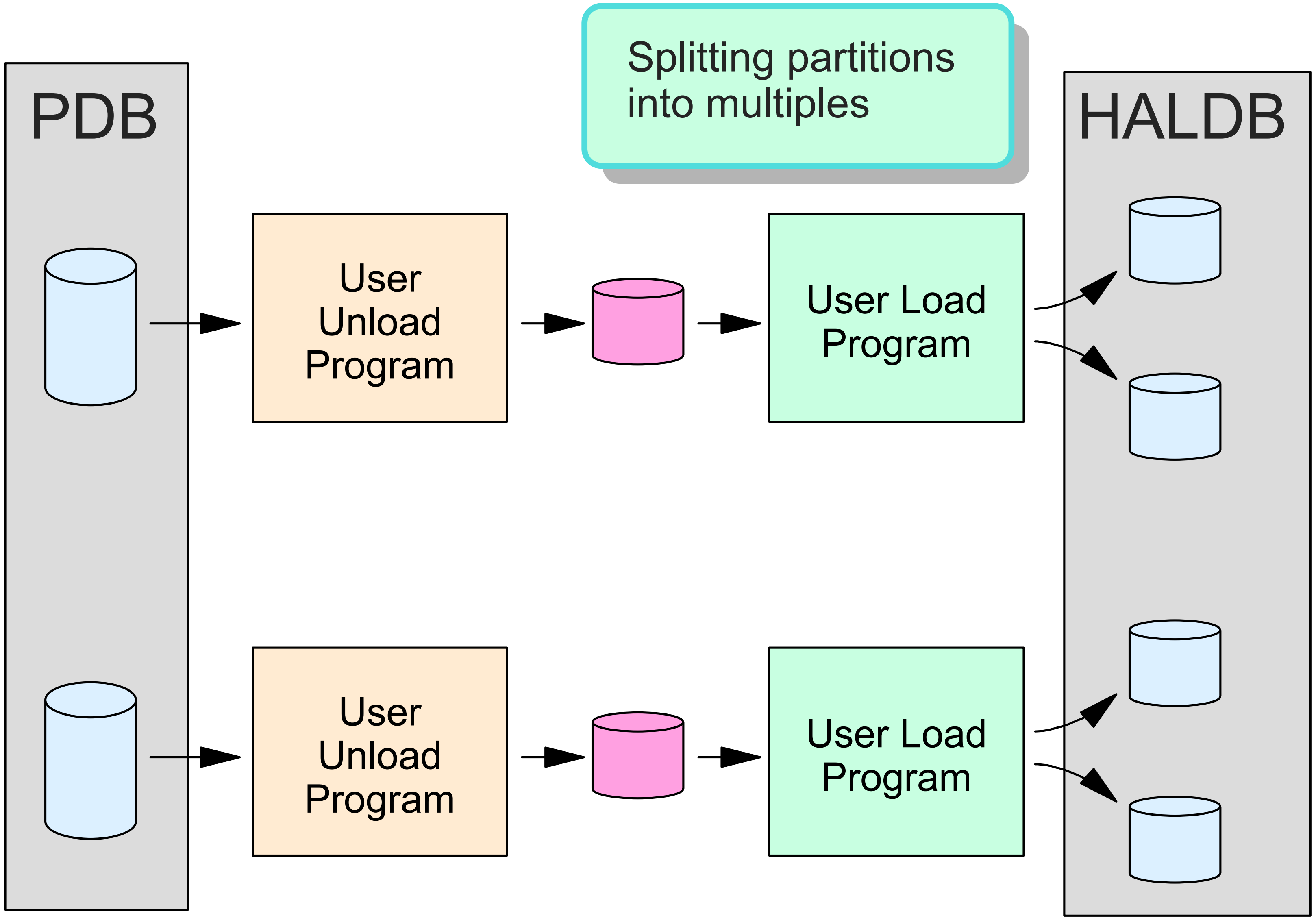
    Load logical parents, then insert logical children using PROCOPT=A

PDB to HALDB - Example 3

Maintaining the same partition boundaries.



PDB to HALDB - Example 4

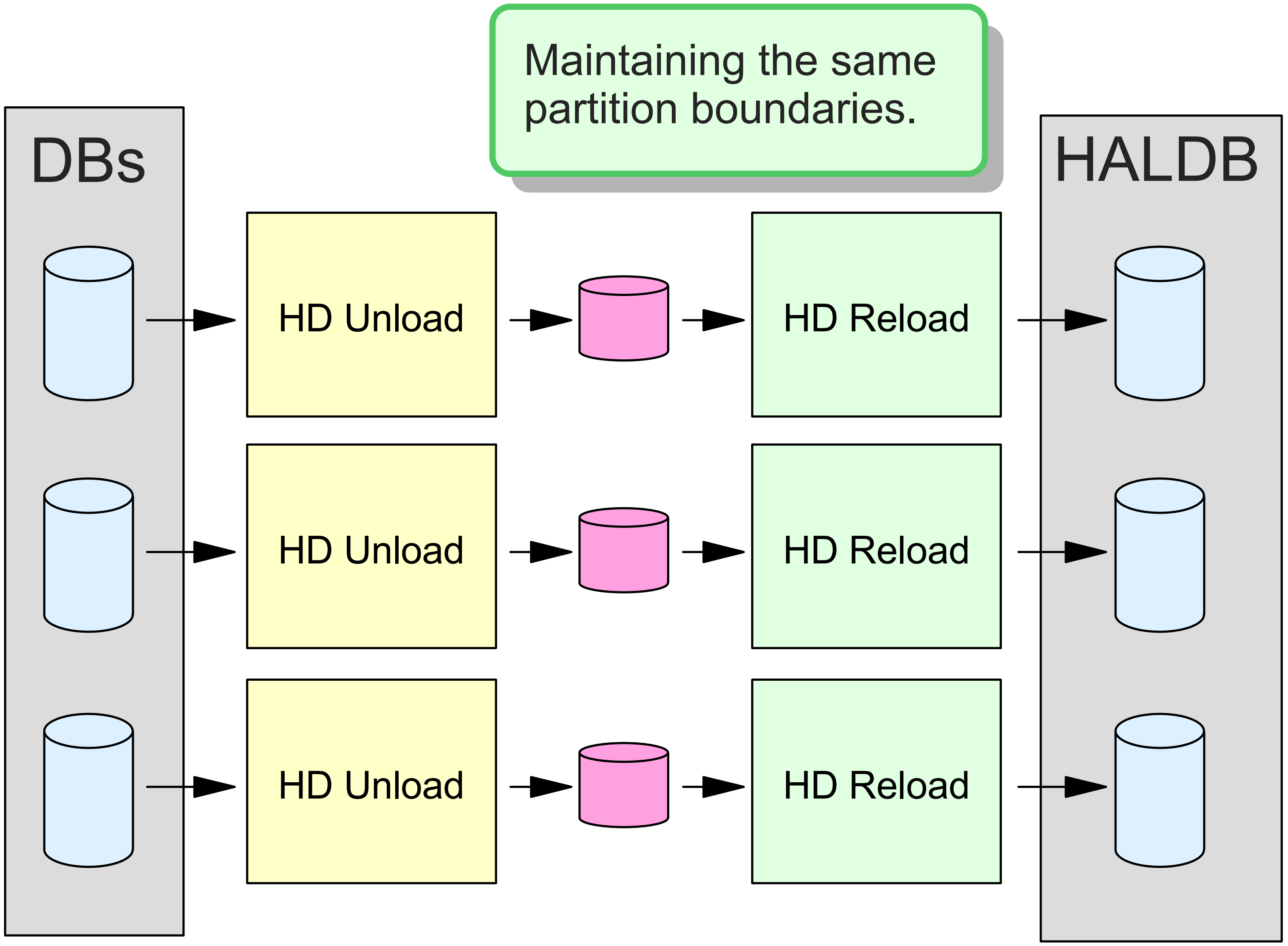


Migrating from User Partitioning

- **Partition boundaries are determined by implementation**

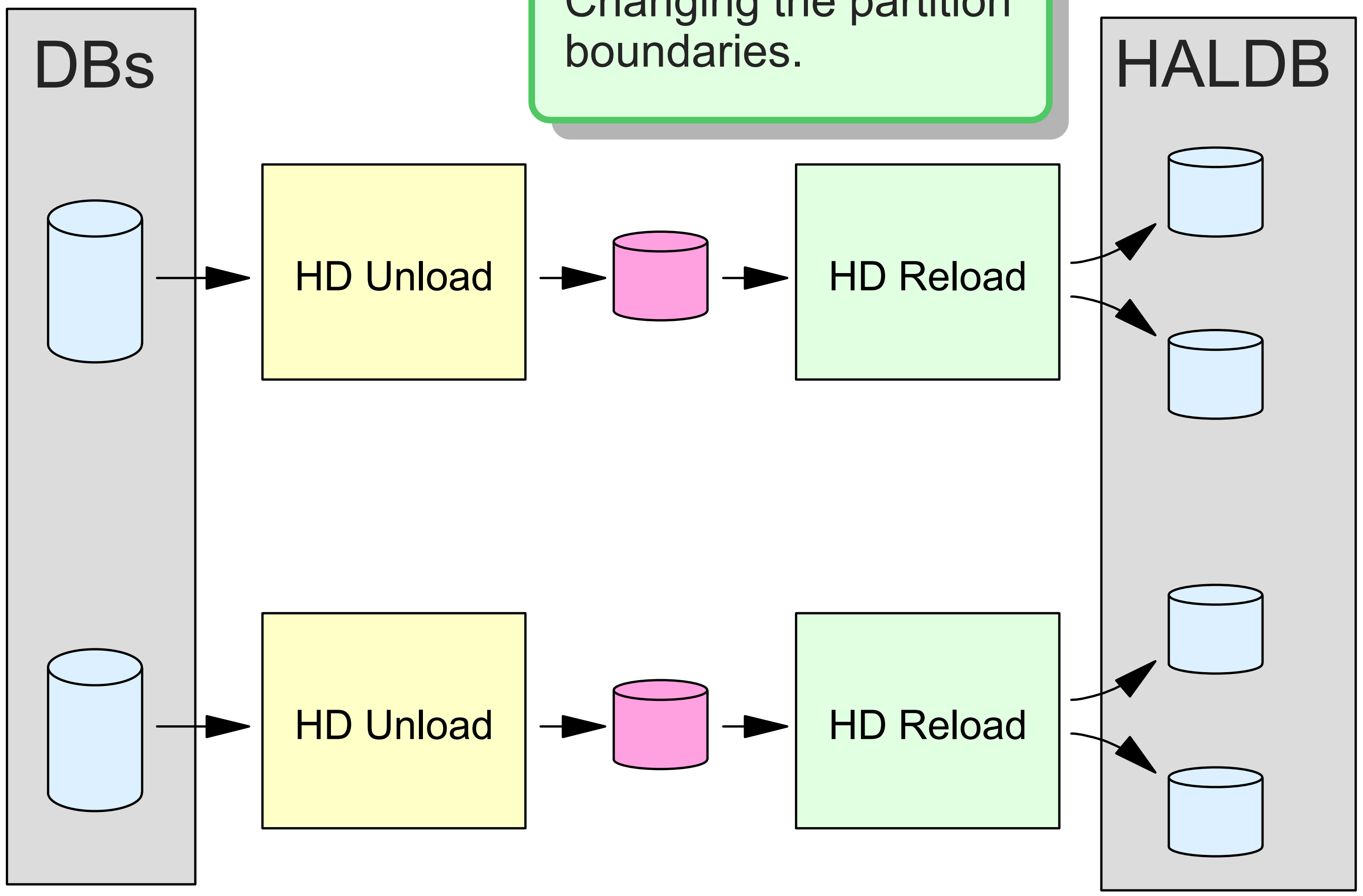
- **Migrating partitions**
  - Each user partition requires execution of Unload and Reload
  - If partition boundaries remain unchanged
    - Each physical database becomes a HALDB partition
    - HD Unload for each user partitioned database
    - HD Reload for each HALDB partition

User Partitioning to HALDB - Example 1



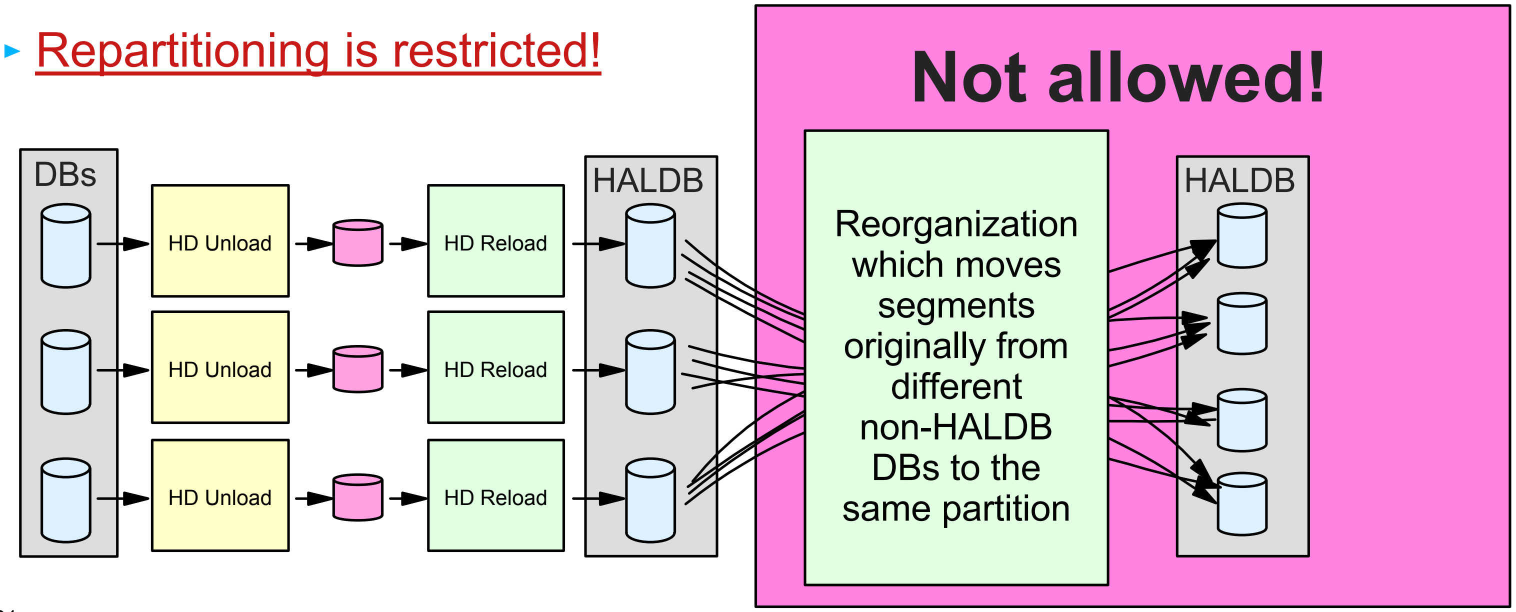
User Partitioning to HALDB - Example 2

Changing the partition boundaries.



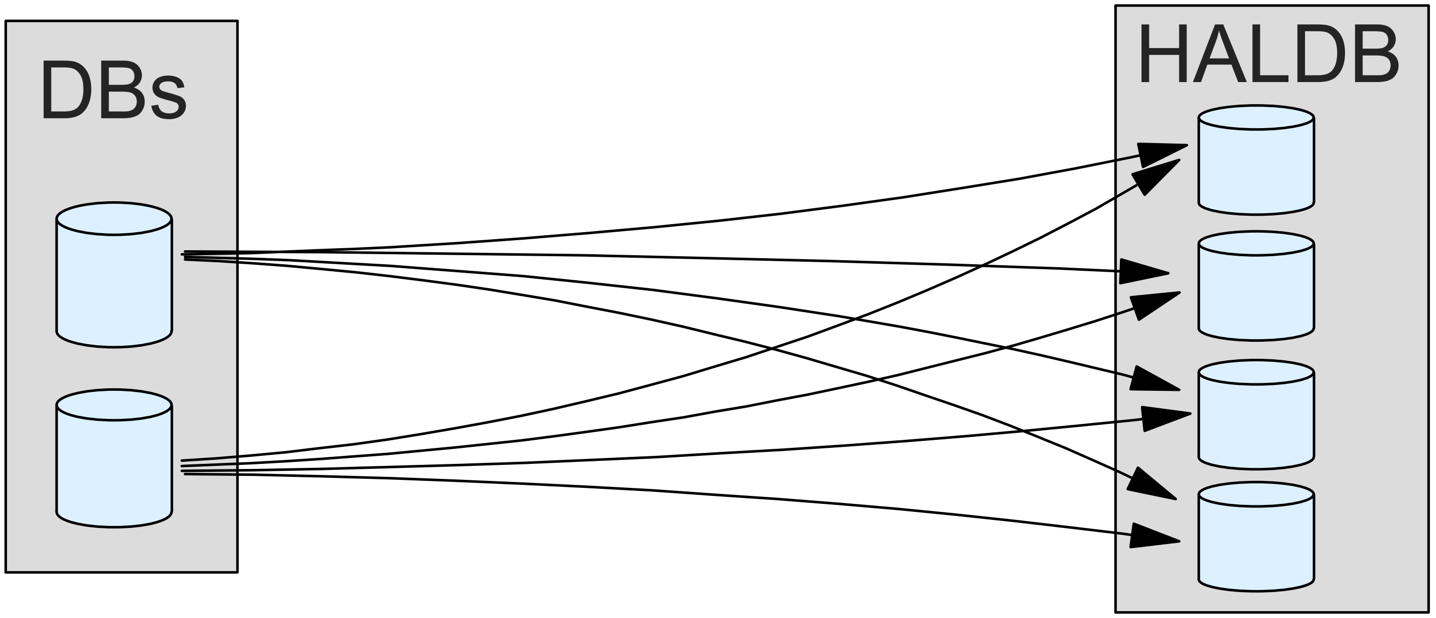
Warning on Repartitioning

- **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!


Not allowed!
DBs
HD Unload
HD Unload
HD Unload
HD Reload
HD Reload
HD Reload
HALDB
Reorganization which moves segments originally from different non-HALDB DBs to the same partition
HALDB

Migrating from User Partitioning

- **Some migrations may require initial load followed by updates of HALDB partitions**
  - Changes of partition boundaries from user partitioning
    - Records from multiple user partitions moved to one HALDB partition



Summary

- **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