HALDB Workshop

HDAM database to PHDAM database conversion - Simple Database structure

Database Structure

- Single Root with several fields
- 5 Secondary Indexes that are not converted but these may be built later using Index Builder
- No Logical relationships
- 5070 segments in the database

Refer to IMS Explorer for a reference view of hierarchy.

The preliminary and set up work has been performed:

HDAM DBD is generated
PSB is generated – note: this will not be changing during the workshop
DBRC Recon datasets are allocated and initialized
HDAM database HDO8 is registered in DBRC
HDAM database HDO8 is allocated and loaded

TSO ID: **IMPOTnn** where nn is your team number
Password: **IBM03POT**

Datasets:

IMS

IMS.IMSB.SDFSRESL ← IMS Reslib dataset
IMS.IMSB.SDFSMAC ← IMS MACLIB dataset
IMS.IMSB.PROCLOAD(DFSVSM0B) ← VSAMP dataset

Static

IMPOTnn.DEMO.RECON1 ← Recon dataset 1
IMPOTnn.DEMO.RECON2 ← Recon dataset 2
IMPOTnn.DEMO.RECON3 ← Recon dataset 3
IMPOTnn.DEMO.MDALIB ← Dynamic Allocation dataset
IMPOTnn.DEMO.HDO8.HDO801 ← HDAM database dataset

Modified

IMPOTnn.DEMO.DBDLIB ← DBDLIB dataset

Allocated

IMPOTnn.DEMO.HDO8.UNLOAD ← HD Unload dataset
IMPOTnn.DEMO.HDO8.UNLOAD.HALDB ← HD Unload migration dataset
IMPOTnn.DEMO.HDO8.HPUNLOAD ← HP Unload dataset
IMPOTnn.DEMO.HDO8.HPUNLOAD.HALDB ← HP Unload migration dataset
IMPOTnn.DEMO.PHDO8.L0000n ← Partition ILDS dataset
IMPOTnn.DEMO.PHDO8.A0000n ← Partition data dataset

JCL

IMPOTnn.DEMO1.JCL

Step 9 Datasets

Static

IMPOTnn.DEMOT.RECON1	← Recon dataset 1
IMPOTnn.DEMOT.RECON2	← Recon dataset 2
IMPOTnn.DEMOT.RECON3	← Recon dataset 3
IMPOTnn.DEMOT.DBDLIB	← DBD dataset
IMPOTnn.DEMOT.HDO8.HDO801	← HDAM database dataset
IMPOTnn WORKSHOP.PGMLIB	← APF library for runtime CSECT

Temporary

IMPOTnn.TEMP.DBDLIB	← DBDLIB dataset
IMPOTnn.TEMP.RECON	← DBDLIB dataset

A return of 00 is expected for each job executed, unless otherwise noted.

When communications is established, enter TSO and use usual logon protocals.

Step 1

Database Analysis

A single OSAM dataset or VSAM Cluster will be converted to at least one OSAM dataset AND one VSAM KSDS. The OSAM dataset will contain the usual database segment data and the VSAM KSDS is the Indirect List Data Set (ILDS). There will be one set of these datasets for each partition defined. (Index datasets may be discussed in a later project)

A decision must be made as to how to convert the HDAM Database to a partitioned database (PHDAM). The outcome of this decision will be the number of partitions defined and the size of these partitioned datasets.

We will use the IMS provided HALDB Migration Utility, DFSMAID0, to provide analysis information used to partition. DFSMAID0 may be executed to provide information in several ways listed below:

NBR - by the number of partitions
KR - by key ranges
MAX - by maximum bytes in each partition
A keyword of SAMPLE may be added to any of the jobs to reduce storage used and the execution time of the database analysis when analyzing larger databases.

The key of our test database is 16 bytes.

The workshop dataset contains several members with DFSMAID0 jobs of various flavors.

Member	Keyword
ONE1	NBR=

Analyze the database for the number partitions entered. You might try different partition values to adjust segment distribution. This value may be up to 1001.

Enter a number of your choice and execute the job.

ONE2 KR=

Analyze the database for specific key ranges. Many key ranges may be entered by placing single KR on each line. KR values may be entered in character or hexadecimal format. A final partition will added to the series of KR statements for High Values. The full length of the actual key does not need be entered, they will be padded with x'FF'.

KR=C'CD' or KR=X'C3C4'

Enter a value or values of your choice and execute the job.

ONE3 KR=

Here we have provided a key range set for each character in the alphabet. Execute this job and note the actual segment distribution for each partition.

ONE4 MAX=

Analyze the database using the actual number of bytes in a partition dataset. Enter a number of bytes to analyze against. This number will apply to all partitions. You might try several different values to adjust the database partitioning. The maximum MAX value is 2147483647.

ONE5 SAMPLE=
NBR=

Analyze the database for a defined number of partitions based on a sample of the segments in the database. The use of SAMPLE provides an estimation of segment distributions that approaches the actual distribution. The User Guide information for DFSMAID0 provides a formula for calculating the accuracy of using the SAMPLE keyword. A small sample may provide key information that is unable to be used successfully. In general, however, the larger the SAMPLE value used, the more accurate the estimation is. You might try different values.

ONE6 NBR=

This is a workshop case I use for 5 partitions. You may vary this if you like. I recommend between 2 and 10 partitions just for initial ease of working and time required.

Save the output from this job. It will used in a later step.

Step 2

HDAM Database Unload

The workshop dataset contains several members with Unload jobs of various flavors:

Member	Unload Type

TWO1 HD Unload

This job is a standard HD Unload without HALDB migration keyword. The output of this job will not be used, but we can browse the content and note the differences with the unload datasets after executing the following job containing MIGRATE=YES.

Execute this job and note the output dataset name.
USERID.DEMO.HD08.UNLOAD

TWO2 HD Unload using MIGRATE=YES

This job is a standard HD Unload using the HALDB migration keyword. The output of this job will be used to load the new HALDB partitions.

Execute this job and note the output dataset name.
USERID.DEMO.HD08.UNLOAD.HALDB

As a comparison, you might browse the contents of each of the output datasets Identified previously to note the differences. The output for a migration contains additional information for building an ILDS during load processing.

TWO3 High Performance Unload

This job is a High Performance Unload without HALDB migration keyword. This job is provided as an comparison of runtime against the HD Unload Utility. There are many instances where a database outage window for conversion to HALDB may not be long enough for the HD Utilities to complete. This comparison is provided to show capabilities not provided in standard IMS utilities. These additional capabilities may provided performance improvements to meet your conversion outage window successfully. The output of this job will not be used, but we can browse the content and note the differences with the unload datasets after executing the following job containing MIGRATE=YES.

Execute this job and note the output dataset name.
USERID.DEMO.HDO8.HPUNLOAD

TWO4 High Performance Unload using MIGRATE

This job is a High Performance Unload without HALDB migration keyword. This job is provided as an comparison of runtime against the HD Unload Utility. There are many instances where a database outage window for conversion to HALDB may not be long enough for the HD Utilities to complete. This comparison is provided to show capabilities not provided in standard IMS utilities. These additional capabilities may

provided performance improvements to meet your conversion outage window successfully. The output of this job will not be used, but we can browse the content and note the differences with the unload datasets after executing the following job containing MIGRATE.

Execute this job and note the output dataset name.
USERID.DEMO.HDO8.HPUNLOAD.HALDB

As a comparison, you might browse the contents of each of the output datasets identified previously to note the differences. The output for a migration contains additional information for building an ILDS during load processing. This information will be quite similar to the comparisons of the HD Unload jobs.

Step 3

Redefine the HDAM DBD as PHDAM. We will use the same DBD name. Using the same DBD name allows following processes to use the same PSB without modification. When this is done, the applications accessing the database are impacted as little as possible.

Member Keyword

THREEDBD

This contains the DBD source for the database HDO8. Create a new member e.g. THREEN by copying member THREEDBD.
This will provide a reference if needed later.

THREEN

Edit this member to be DBD source for a PHDAM Database. There are minor modifications for this DBD.

- Change HDAM to PHDAM
- Remove the DATASET statement completely.
-

before

```
         DBD   NAME=HDO8,ACCESS=(HDAM,OSAM),
               RMNAME=(DFSHDC40,5,120)
*
         DATASET DD1=HDO801,DEVICE=3390,SIZE=4096
*
         SEGM  NAME=ROOT,BYTES=(50)
*              COMPRTN=(DFSCMPX0,DATA)
```

after

```
         DBD   NAME=HDO8,ACCESS=(PHDAM,OSAM),
               RMNAME=(DFSHDC40,5,120)
*
*        DATASET DD1=HDO801,DEVICE=3390,SIZE=4096
*
         SEGM  NAME=ROOT,BYTES=(50)
*              COMPRTN=(DFSCMPX0,DATA)
```

THREE1

This is a DBDGEN job that when executed will replace the existing DBD in your DBDLIB. You probably will want to rename the current DBD prior to executing the job for fallback purposes. Verify the the input member value for the parm in the execution statement MBRA= matches the name of your modified DBD source member. This has been preset to be THREEN.

Execute this job.

Step 4

DBRC Partition Definitions for registration

The information input here will be based on the analysis information from the ONE6 job executed from Step 1 Member ONE6

Each partition requested by the DFSMAID0 run in job ONE6 will print information as below.

```
partition 3 :

   minimum key =

      +0000  d550e6e6 c5d950e8 e4c9d6d7 d8e6c5d9  |N&WWER&YUIOPQWER|

   maximum key =

      +0000  f0f0e6f2 865b84a2 f7f8f9f0 f1f2f3f4  |00W2f$ds78901234|

                      segments         bytes   prefix-incr   length-incr
   1) 'ROOT    '          1014         56784          8112             0
 SUM)                      1014         56784          8112             0
```

minimum or LOW key value
maximum or HIGH key value

and information regarding any length of segment increase. This a total length increase for the partition that would be used to allow more space than in the original dataset when you allocate the new partition dataset.

The maximum key values will be the most important here. They will be used to define the partitions to DBRC. Do not delete this job. You will refer to it in some of the following steps.

Member

FOUR1
This job will perform a Delete of the HDO8 database from the DBRC Recon.

Execute this job and verify in the listing that the database is no longer registered.

FOUR2
This job will register a HALDB database named HDO8 and the number of partitions you decided on from the job run as ONE6. You will need one INIT.PART statement for each partition you will be defining. In the sample below.....

```
INIT.PART DBD(HDO8) PART(HDO80nn) -
          DSNPREFX(USERID.DEMO.PHDO8) -
          KEYSTRNG(xxxxxxxxxxxxxxxx) GENMAX(5)
```

A – replicate the INIT.PART to match the number of partitions

B – change the n on the PART(HDO80 for each partition to an ascending sequential number. 1, 2, nnn

C - in the KEYSTRING (highlighted above) enter the value of the maximum key from each partition from the DFSMAID0 job run as ONE6 previously. I highlighted a sample above. Each partition will have increasingly higher KEYSTRING values.
Note 1: The DBRC parser has some limitations. When lower case or some special characters are used as part of the partition key these characters must be entered as hexadecimal values. This is shown in the example below.
Note 2: It is acceptable to enter the KEYSTRNG value as a subset of the entire key. The value entered will be left justified and the remaining length of the key padded with x'FF'

e.g.
KEYSTRNG(ABC123)
will become
KEYSTRNG(X'C1C2C3F1F2F3FFFFFFFFFFFFFFFFFFFF')

Here is an example:

```
INIT.PART DBD(HDO8) PART(HDO8001) -
          DSNPREFX(USRT001.DEMO.PHDO8) -
          KEYSTRNG(X'F0F0E6F2865B84A2F7F8F9F0F1F2F3F4') -
          GENMAX(5)
```

HINT:

– Paste in the information from the MAX String of job ONE6 into this job and delete the blanks

Make sure that this maximum key value
being cut/pasted

```
partition 1 :

   minimum key =

      +0000  40404040 40404040 4040d6d7 d8e6c5d9  |         OPQWER|

   maximum key =

      +0000  a39392e6 5050d9e3 e8e4c9d6 d7d8e6c5  |tlkW&&RTYUIOPQWE|

                     segments        bytes   prefix-incr   length-incr
   1) 'ROOT    '         1014        56784          8112             0
 SUM)                    1014        56784          8112             0

-------------------------------------------------------------------------
```

into here on line 22

```
EDIT       IMPOT01.DEMO1.JCL(FOUR2) - 01.20              Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
000019  INIT.DB DBD(HDO8) SHARELVL(1) TYPHALDB OLRCAP
000020  INIT.PART DBD(HDO8) PART(HDO8001) -
000021            DSNPREFX(IMPOT01.DEMO.PHDO8) -
000022            KEYSTRNG(X'a39392e6 5050d9e3 e8e4c9d6 d7d8e6c5-
000023            GENMAX(5)
000024  INIT.PART DBD(HDO8) PART(HDO8002) -
000025            DSNPREFX(IMPOT01.DEMO.PHDO8) -
000026            KEYSTRNG(X'D550E6E6C550E3E8E4C9D6D7D8E6C5D9') -
```

gets edited to this on line 22

No Blanks
No lower case characters

```
EDIT       IMPOT01.DEMO1.JCL(FOUR2) - 01.21              Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
000019  INIT.DB DBD(HDO8) SHARELVL(1) TYPHALDB OLRCAP
000020  INIT.PART DBD(HDO8) PART(HDO8001) -
000021            DSNPREFX(IMPOT01.DEMO.PHDO8) -
000022            KEYSTRNG(X'A39392E65050D9E3E8E4C9D6D7D8E6C5') -
000023            GENMAX(5)
000024  INIT.PART DBD(HDO8) PART(HDO8002) -
000025            DSNPREFX(IMPOT01.DEMO.PHDO8) -
000026            KEYSTRNG(X'D550E6E6C550E3E8E4C9D6D7D8E6C5D9') -
000027            GENMAX(5)
000028  INIT.PART DBD(HDO8) PART(HDO8003) -
000029            DSNPREFX(IMPOT01.DEMO.PHDO8) -
000030            KEYSTRNG(X'F0F0E6F2865B84A2F7F8F9F0F1F2F3F4') -
000031            GENMAX(5)
```

NOTE:

- The HEX string MUST be in capitals, the DBRC parser requires it
- Make sure the quotes are closed on the **`KEYSTRNG`** statements
- Make sure each **PART** values increments

execute this job and verify that the LIST.RECON shows the new HALDB database and partition information.

You should note that the PARTITION INIT NEEDED is set to YES. This will be addressed in Step 6 after the datasets have been allocated.

Step 5

Member	Keyword
FIVE1	

This job will delete and define the database datasets for the number of partitions you have chosen..

Each partition has two datasets: an OSAM component for data and a VSAM KSDS for the ILDS. You will need to define a pair datasets for each partition you are using. The job is set up in three steps. The first step is an IDCAMS delete of all datasets. The second step is an IEFBR14 to allocate the OSAM components. The third step is an IDCAMS allocate of the VSAM components. Replicate the supplied JCL statements to the equivalent number of partitions you chose and edit the control cards for the partition names being allocated. It is a good idea to replicate the delete information to match also.

e.g.

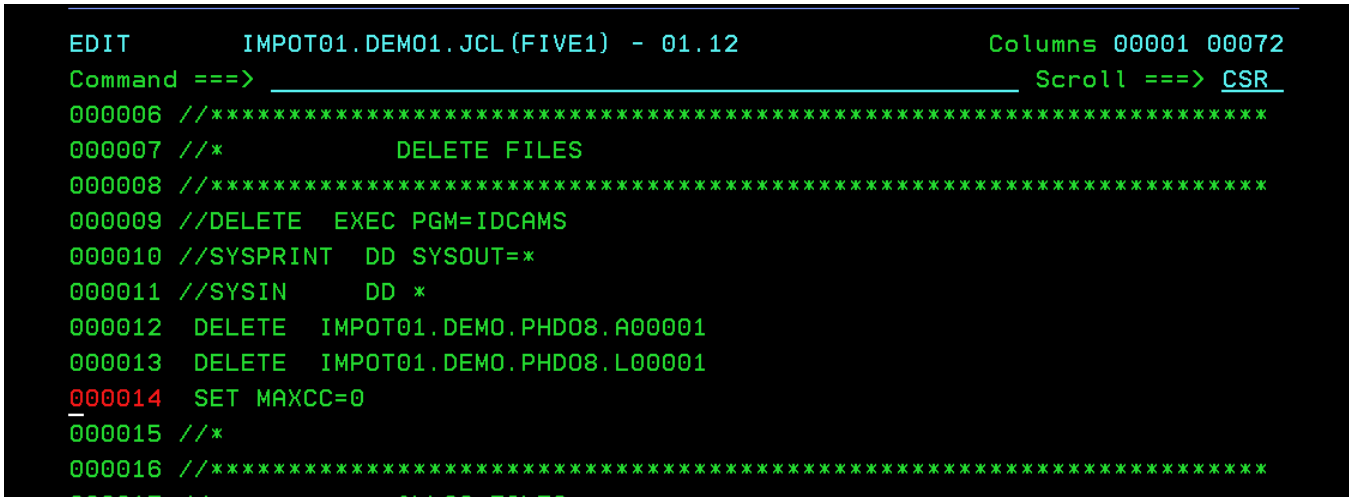
```
//PHDO8A1   DD DSN=DDS0027.DEMON.PHDO8.A00001,
//             DISP=(,CATLG),UNIT=3390,VOL=SER=DMEU07,
//             SPACE=(CYL,(2,1))
```

```
e.g.
//SYSIN   DD  *
 DEFINE CLUSTER(NAME(DDS0027.DEMON.PHDO8.L00001) -
       INDEXED KEYS(9,0), RECSZ(50,50)           -
       REUSE SPEED                               -
       CISZ(4096) VOL(DMEU07) CYL(1,1) -
       )
```

The highlighted number above will be incremented for each partition you have.

In the case of using 5 partitions

This



becomes

```
EDIT       IMPOT01.DEMO1.JCL(FIVE1) - 01.11               Columns 00001 00072
Command ===> _                                             Scroll ===> CSR
000006 //*********************************************************************
000007 //*           DELETE FILES
000008 //*********************************************************************
000009 //DELETE  EXEC PGM=IDCAMS
000010 //SYSPRINT  DD SYSOUT=*
000011 //SYSIN     DD *
000012  DELETE  IMPOT01.DEMO.PHDO8.A00001
000013  DELETE  IMPOT01.DEMO.PHDO8.L00001
000014  DELETE  IMPOT01.DEMO.PHDO8.A00002
000015  DELETE  IMPOT01.DEMO.PHDO8.L00002
000016  DELETE  IMPOT01.DEMO.PHDO8.A00003
000017  DELETE  IMPOT01.DEMO.PHDO8.L00003
000018  DELETE  IMPOT01.DEMO.PHDO8.A00004
000019  DELETE  IMPOT01.DEMO.PHDO8.A00005
000020  DELETE  IMPOT01.DEMO.PHDO8.L00004
000021  DELETE  IMPOT01.DEMO.PHDO8.L00005
000022  SET MAXCC=0
000023 //*
000024 //*********************************************************************
```

This Allocation

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       IMPOT01.DEMO1.JCL(FIVE1) - 01.12               Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
000017 //*           ALLOC FILES
000018 //*********************************************************************
000019 //ALLOC  EXEC  PGM=IEFBR14
000020 //PHDO8A1   DD DSN=IMPOT01.DEMO.PHDO8.A00001,
000021 //             DISP=(,CATLG),UNIT=3390,VOL=SER=DMEU07,
000022 //             SPACE=(CYL,(2,1))
000023 //*
000024 //ALLOC   EXEC PGM=IDCAMS
```

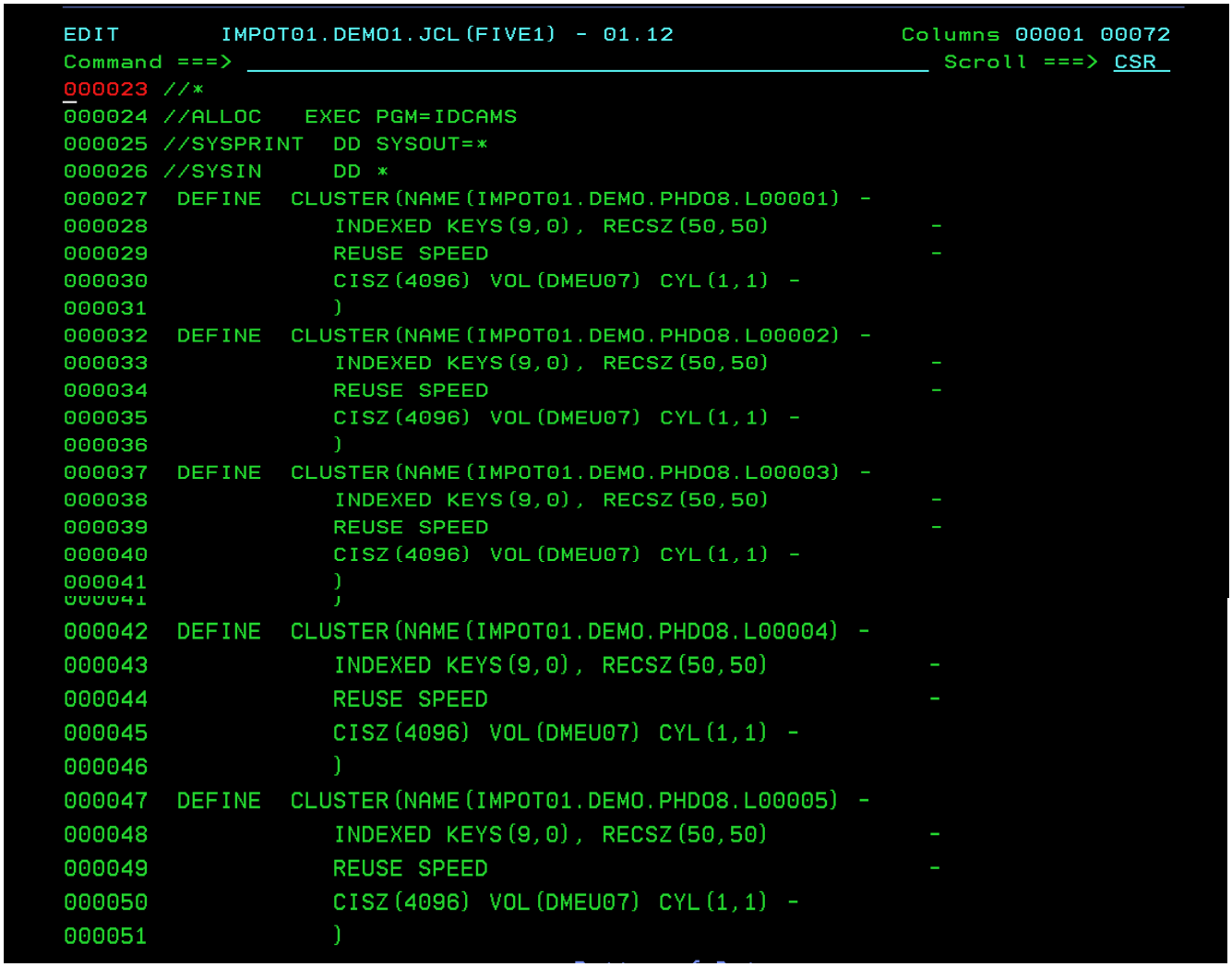
becomes this

```
EDIT       IMPOT01.DEMO1.JCL(FIVE1) - 01.12               Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
000016 //*******************************************************************
000017 //*           ALLOC FILES
000018 //*******************************************************************
000019 //ALLOC  EXEC  PGM=IEFBR14
000020 //PHDO8A1   DD DSN=IMPOT01.DEMO.PHDO8.A00001,
000021 //             DISP=(,CATLG),UNIT=3390,VOL=SER=DMEU07,
000022 //             SPACE=(CYL,(2,1))
000023 //PHDO8A2   DD DSN=IMPOT01.DEMO.PHDO8.A00002,
000024 //             DISP=(,CATLG),UNIT=3390,VOL=SER=DMEU07,
000025 //             SPACE=(CYL,(2,1))
000026 //PHDO8A3   DD DSN=IMPOT01.DEMO.PHDO8.A00003,
000027 //             DISP=(,CATLG),UNIT=3390,VOL=SER=DMEU07,
000028 //             SPACE=(CYL,(2,1))
000029 //PHDO8A4   DD DSN=IMPOT01.DEMO.PHDO8.A00004,
000030 //             DISP=(,CATLG),UNIT=3390,VOL=SER=DMEU07,
000031 //             SPACE=(CYL,(2,1))
000032 //PHDO8A5   DD DSN=IMPOT01.DEMO.PHDO8.A00005,
000033 //             DISP=(,CATLG),UNIT=3390,VOL=SER=DMEU07,
000034 //             SPACE=(CYL,(2,1))
                                                        05/015
```

And this allocation

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       IMPOT01.DEMO1.JCL(FIVE1) - 01.12               Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> CSR
000023 //*
000024 //ALLOC   EXEC PGM=IDCAMS
000025 //SYSPRINT  DD SYSOUT=*
000026 //SYSIN     DD *
000027  DEFINE  CLUSTER(NAME(IMPOT01.DEMO.PHDO8.L00001) -
000028             INDEXED KEYS(9,0), RECSZ(50,50)          -
000029             REUSE SPEED                              -
000030             CISZ(4096) VOL(DMEU07) CYL(1,1) -
000031             )
****** **************************** Bottom of Data ****************************
```

becomes this



execute this job

Step 6

The HALDB now needs to be initialized. There are two ways of doing this. Using the partition initialization utility DFSUPNT0 or using the IMS database pre-reorganization utility DFSURPR0. Either method can be used here. Run member SIX1 for DFSUPNT0. Run SIX2 for DFSURPR0. There is no need to run both. Surprise me. Pick one.

Member	Keyword

SIX1

This job will initialize the HALDB using DFSUPNT0. The utility may be run as a stand-alone job as we are doing here, or as a ULU region. DFSUPNT0 may be used to initialize several HALDBs at one time. Multiple HALDBs are entered each on a single line. Here we are only initializing HDO8.

execute this job

SIX2

This job will initialize the HALDB using DFSURPR0. The utility may be run as a stand-alone job as we are doing here, or as a ULU region. DFSURPR0 may be used to initialize several HALDBs at one time. Multiple HALDBs are entered separated by commas. Each entry is expected to be 8 bytes long, so any database name less that 8 must be left justified and filled with blanks.

The control card would look like:

DBIL=HALDB1 ,HALDBONE,LASTDBD

Here we are only initializing HDO8. DBIL=HDO8

execute this job – if SIX1 is not being used.

SIX3

This job will turn off the Image Copy needed flag in the DBRC Recon. If the flag is left on the Load job will not be able to obtain DBRC authorization to allocate the database datasets. Replicate each CHANGE.DBDS statement for each partition defined previously in job FOUR2.

e.g.

```
CHANGE.DBDS DBD(HDO8001) DDN(HDO8001A) ICOFF  ← for each partition
CHANGE.DBDS DBD(HDO8002) DDN(HDO8002A) ICOFF
```

Execute this job.

Step 7

Member	Keyword

SEVEN1

This is the load job using HD Reload – DFSURGL0. In this job we are using the the DFSURCDS and DFSURWF1 files for compatibility. The input for loading the database is the output from the unload job in step 2 Member TWO2

Execute this job. Verify that the number of segments loaded matches the number of segments that were unloaded in Step TWO2.

Note:

There will be messages issued noting that the IMAGE COPY NEEDED number will have been incremented in DBRC Recons has been turned on for each of the partition. This is normal.

Congratulations: A successful migration to HALDB has been completed. In order to use the database

an Image Copy should be completed prior to starting the database.

Step 8 – Extra Credit

Member	Keyword
EIGHT1	

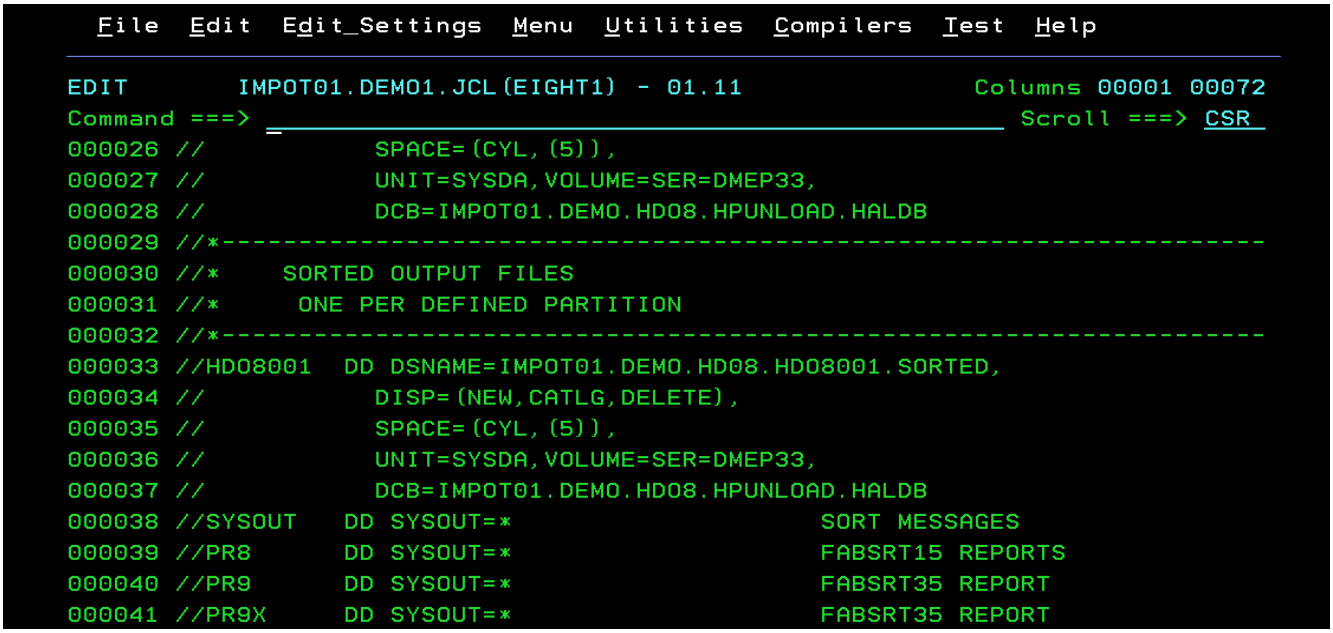
This a PSSR sort job for the input to a HALDB using the HP Unload output. If a PSSR job is not run prior to attempting a Load using HP Load then an Abend U3726 is received during the HP Load process. This job will sort the segments into individual partition input files.

In this JCL note the DD names that correspond to the DD names of the partitions you previously defined to DBRC as part of the registration process. The input for loading the database is the output from the unload job in step 2 Member TWO4. The output sorted into each DD will be used as input to the HP Load job.

Look for the section in the JCL with comments of DD cards here. Add the DD names for any partition that was defined in the previous steps. The first DD has the DCB attributes assigned. Carry these attributes to any DD names you add to the JCL.

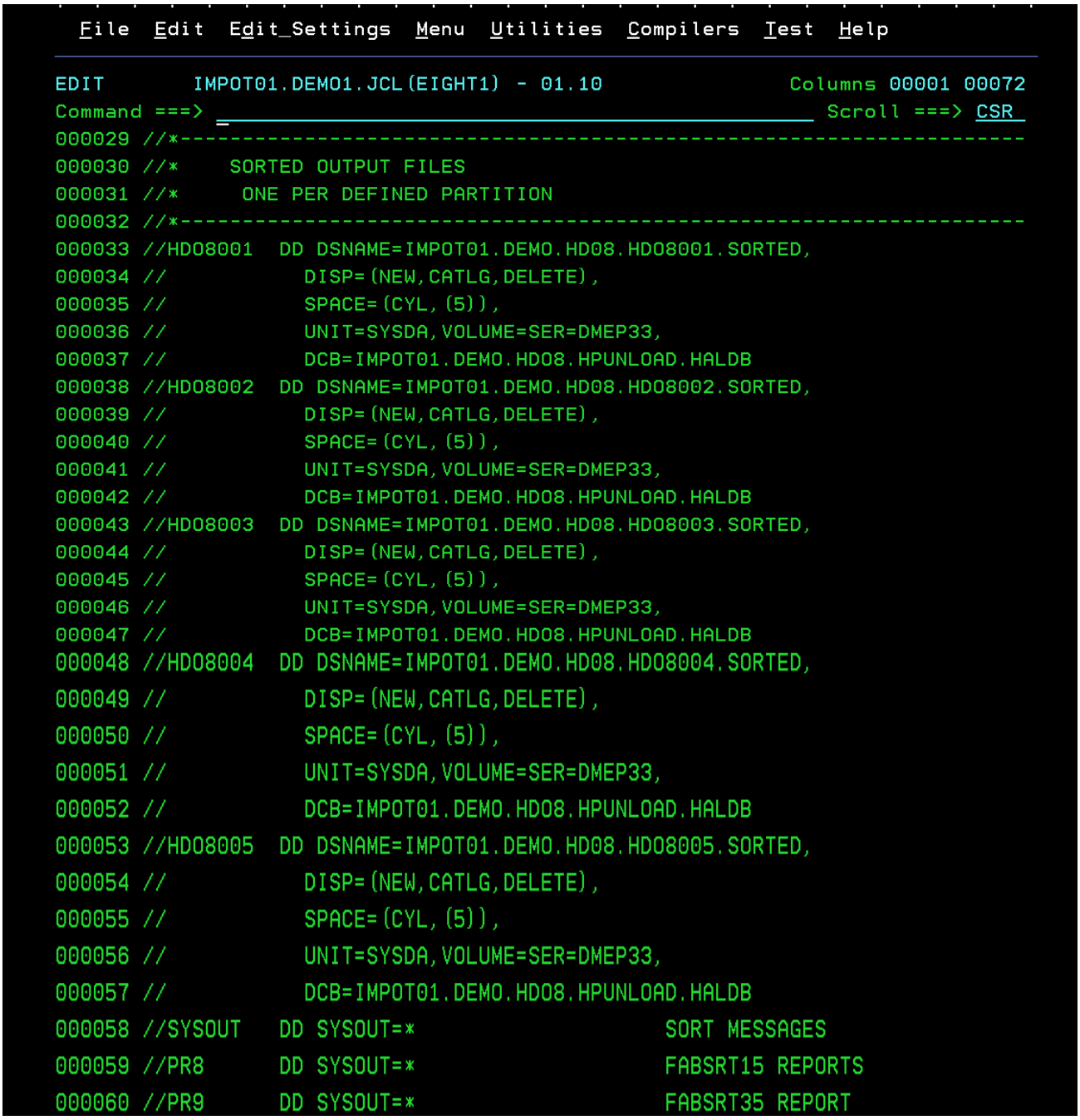
The job EIGHT1 changes are from

this



The SORT values do not need to be changed

to this



FIVE1

This job will delete and define the database datasets.

Rerun this job to create a new set of partition datasets. This job was set up previously and needs no changes.

SIX1

This job will initialize the HALDB using DFSUPNT0.

Rerun this job to initialize the new set of partition datasets. This job was set up

previously and needs no changes.

Re run this job

SIX3

This job will turn off the Image Copy needed flag in the DBRC Recon. If the flag is left on the Load job will not be able to obtain DBRC authorization to allocate the database datasets. Replicate each CHANGE.DBDS statement for each partition defined previously in job FOUR2.

e.g.

```
CHANGE.DBDS DBD(HDO801) DDN(HDO801A) ICOFF  ← for each partition
CHANGE.DBDS DBD(HDO802) DDN(HDO802A) ICOFF
```

Execute this job.

EIGHT2

This is the load job using HP Reload – DFSURGL0. In this job we are using the the DFSURCDS and DFSURWF1 files for compatibility. The input for loading the database is the output from the PSSR job in Step 8 member EIGHT1

Execute this job.

The following three jobs are presented with the assumption that a HALDB with 5 partitions is being used.

EIGHT21

This is the load job using HP Reload – This is the same program as in EIGHT2. The input is now in a single DFSUINPT DD with all of the sorted files from job EIGHT1 concatenated. The concatenated files do not need to be in order.

Execute this job.

EIGHT22

This is the load job using HP Reload – This is the same program as in EIGHT2. This job is set up as a single step for each partition that is being loaded. Each step has a single sorted file form job EIGHT1 as input in DFSUINP DD..

Execute this job.

EIGHT23

This is the load job using HP Reload – This is the same program as in EIGHT2. The input is now in a separate jobs, one per each partition dataset to be loaded. All jobs will run in parallel.

Execute this job.

Step 9 – More Extra Credit

This step uses the IMS HALDB Toolkit to convert our database. The HALDB Toolkit has several additional functions that are not directly addressed in this workshop.

Member	Keyword

NINE1

This an IMS HALDB Toolkit set up job. This job will set up defaults for the HALDB conversion process. performed in

There are three values to be aware of.

RECON – temporary Recon dataset name for conversion

DBDSN – temporary DBD dataset name for conversion

HPUTIL – Load library of utilities if not IMS standard utility

These have been filled in for you.

Execute this job

NINE2

This an IMS HALDB Toolkit job. This job will perform all steps needed to convert our database HDO8 to a HALDB of the same name in a single job. Every step performed in the previous series on ONE6 to SEVEN1 will be performed in a single job step.

Prior to executing this job, the decision of how to partition must be made. This we have previous done in Step 1. Since we have arbitrarily chosen 5 partitions in the previous steps, we will continue using 5 partitions to be consistent, as the new HALDB structure.

There are several keywords to be aware of:

```
CONVERT  DBD(HDO8)   -          <== Database being converted to HALDB
    DBDPATT(*****...) -
    INDPATT(**.***..) -
    FIRSTPART(001)    -          <== First partition number
    ONLINE(N) -
    PARTNUM(5) -                 <== Number of partitions to create
    DSNPREF(IMPOT01.DEMO) -      <== Dataset name prefix
    ICOFF(N)  -                  <== turn off Image Copy needed
    TAKEOVER(Y) -
    VOLALLO(1,2,2) -
    OVFLINCR(300) -
    ICHLQ(DDS0027.DEMO.IC) -
    ICTRLR(2) -
    DATACLAS(*)
```

DBD – the name of the DBD to be converted
DSNPREF – the hlq of the partition DBDS names
ICHLG – the hlq of the Image Copy datasets

this information has been pre filled into the control cards.

Execute this job