

Understanding your IMS Catalog

Deepak Kohli
deepakk@us.ibm.com
IMS Product Manager

CCDUG 2022 Conference
September 20, 2022

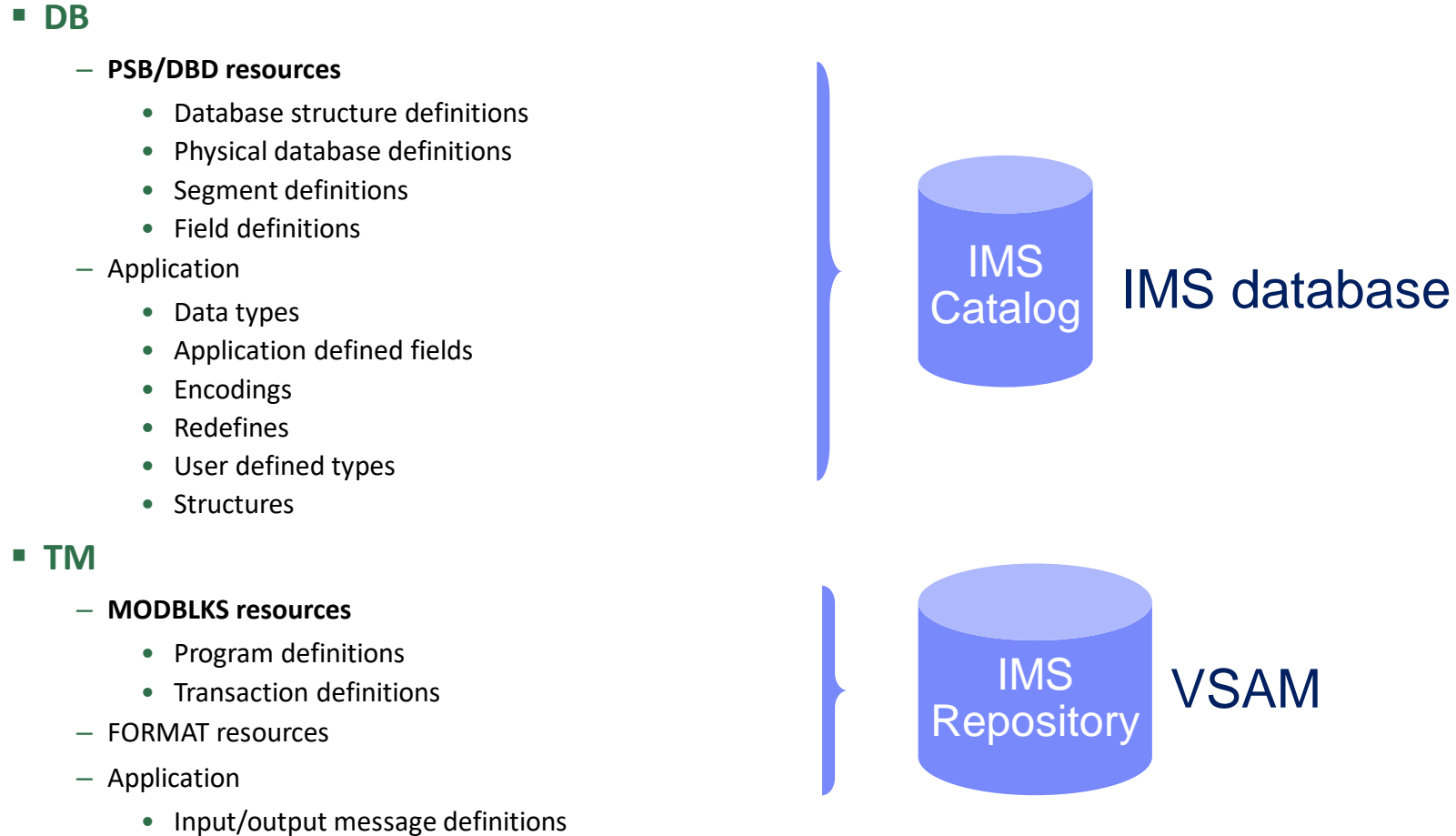


- Clear the confusion
- IMS Catalog Overview
- IMS Catalog Implementations / Configurations
- IMS Catalog Enablement - Reference
- Housekeeping for the Catalog



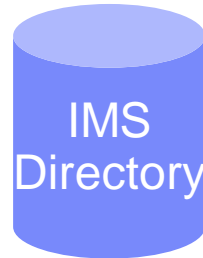
The 3 Stooges – IMS Catalog, Directory & Repository







DBD, PSB metadata used by
Java programs
(in a HALDB)



ACB (DBD, PSB) used by
IMS – Runtime blocks
(in PDSE)

IMS Catalog Overview



What is the IMS Catalog?

What is the IMS Catalog?



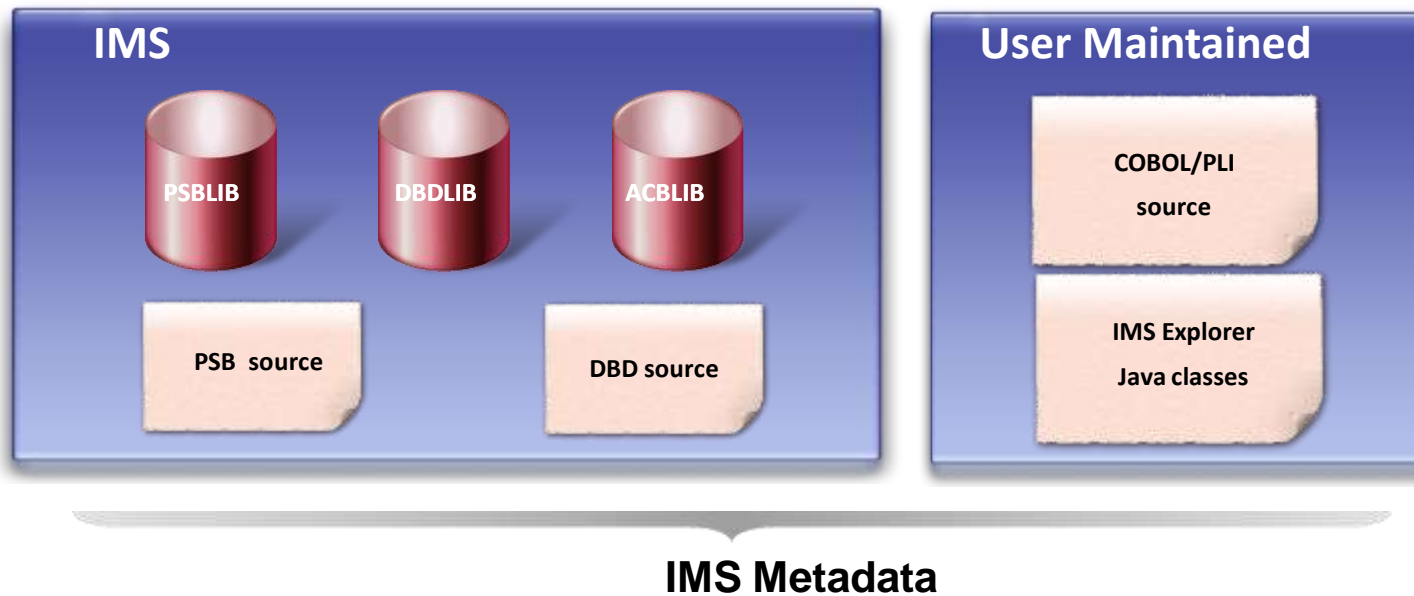
- *Introduced in IMS V12*

- *It's a system database*
 - High Availability Large Database (HALDB)

- **It contains IMS Database & Application metadata.**
 - **What metadata:**
 - **DBDs, COBOL Copybooks, PL/1 Includes, PSBs**

***Where was this metadata
before the IMS Catalog?***

- Databases partially defined in the IMS DBD
 - Only key/searchable fields needed by applications
 - Remaining segment data is not defined
- Remaining database definition is within Applications
 - COBOL COPYBOOKS and PL/I INCLUDES map all the segment data
 - Applications can have different mappings for one segment



Who uses this metadata?

- Leveraged by Universal JDBC Drivers for metadata exchange and discovery
 - The metadata is used by the IMS Universal JDBC driver to allocate program specification blocks (PSBs), issue DL/I calls, perform data transformation, and translate SQL queries to DL/I calls
 - Basically used for Java programs accessing IMS databases

- Going forward the Catalog will also be used for DDL



- IBM Products which can use the Catalog:
 - IMS Explorer for Development
 - Cognos 10.2 is certified to use the Catalog
 - InfoSphere Data Architect
 - IBM Data Studio
 - Rational Asset Analyzer
 - IBM MobileFirst (formerly IBM Worklight Foundation)
 - InfoSphere DataStage
 - InfoSphere Metadata Asset Manager (IMAM)

Can I see what's in the Catalog?

- Accessible via:
 - JDBC/SQL calls
 - Native SQL (COBOL/SQL introduced in MS 13) and
 - DL/I calls

- Applications (and tooling) can access the online IMS Catalog (for example E4D)

Are there cases where Catalog is required?

When is Catalog required?



- Accessing IMS data from Java programs

- IMS features:
 - Database Versioning
 - IMS Native SQL support for COBOL
 - .NET access to IMS data
 - IMS Managed ACBs
 - DDL (requires IMS Managed ACBs, which requires the IMS Catalog)

What are the Catalog DBDs & PSBs?

- **IMS PHIDAM/OSAM HALDB database**
 - Defined with 4 DSGs (Data Set Groups)
- **Has one Secondary Index**
 - Can be used to determine which IMS programs (PSBs) reference a specific user database without processing the entire IMS catalog
- **Unique features**
 - **DBRC use is optional** for the IMS Catalog HALDB database
 - ONLY HALDB that isn't required to be defined in the DBRC RECONS
 - IMS can manage allocation/creation of catalog database data sets
 - Uses parameters in the "CATALOG" section of DFSDFxxx PROCLIB member



- IMS provides DBD and PSB **source code** for the Catalog database

- IMS provides **object code** for the Catalog DBDs and PSBs

- PHIDAM DBD reserved name is **DFSCD000**

- PSINDEX DBD reserved name is **DFSCX000**
 - Used to connect DBDs to PSBs that reference them

- PSBs provided to load, read and update the Catalog database
 - **DFSCPL00** is used for **initial load** process
 - Used by the Catalog Populate Utility

 - **DFSCP001** is used for **update** access
 - Used by ACBGEN and Catalog Populate Utility, and Purge Utility

 - **DFSCP000** - used for read access from COBOL/HLASM programs

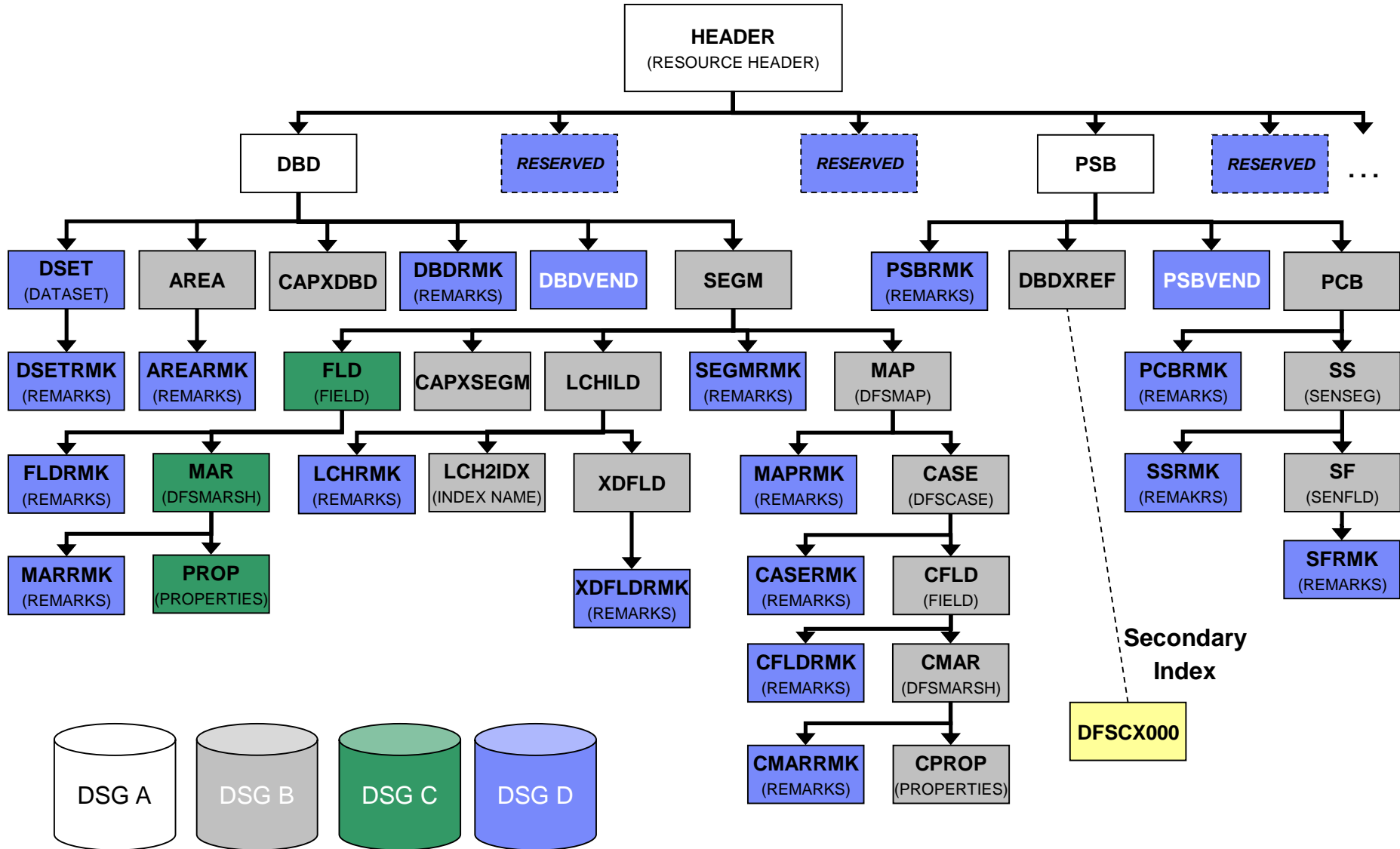
 - **DFSCP002** - used for read access from PL/1 programs

 - **DFSCP003** - used for read access from PASCAL programs

- IMS automatically attaches PCBs for the IMS catalog to each user PSB at run time (when the Catalog is enabled in an IMS system)

What is the Catalog Hierarchy?

Physical Catalog Structure



- Root segment of the Catalog database is a generic resource header
 - Indicates the type of resource → DBD or PSB
 - A dependent of the Root and its Children are a complete DBD or PSB
 - Multiple iterations/instances of a specific resource are supported
 - Most resources are differentiated by their ACBGEN timestamp
 - Logical DBDs and GSAM are differentiated by their DBDGEN timestamp
- Catalog database segments typically correspond to macro statements in the DBD and PSB source
- One segment at the first Child level under both the DBD and PSB segments is available for vendor/customer use
 - DBDVEND
 - PSBVEND

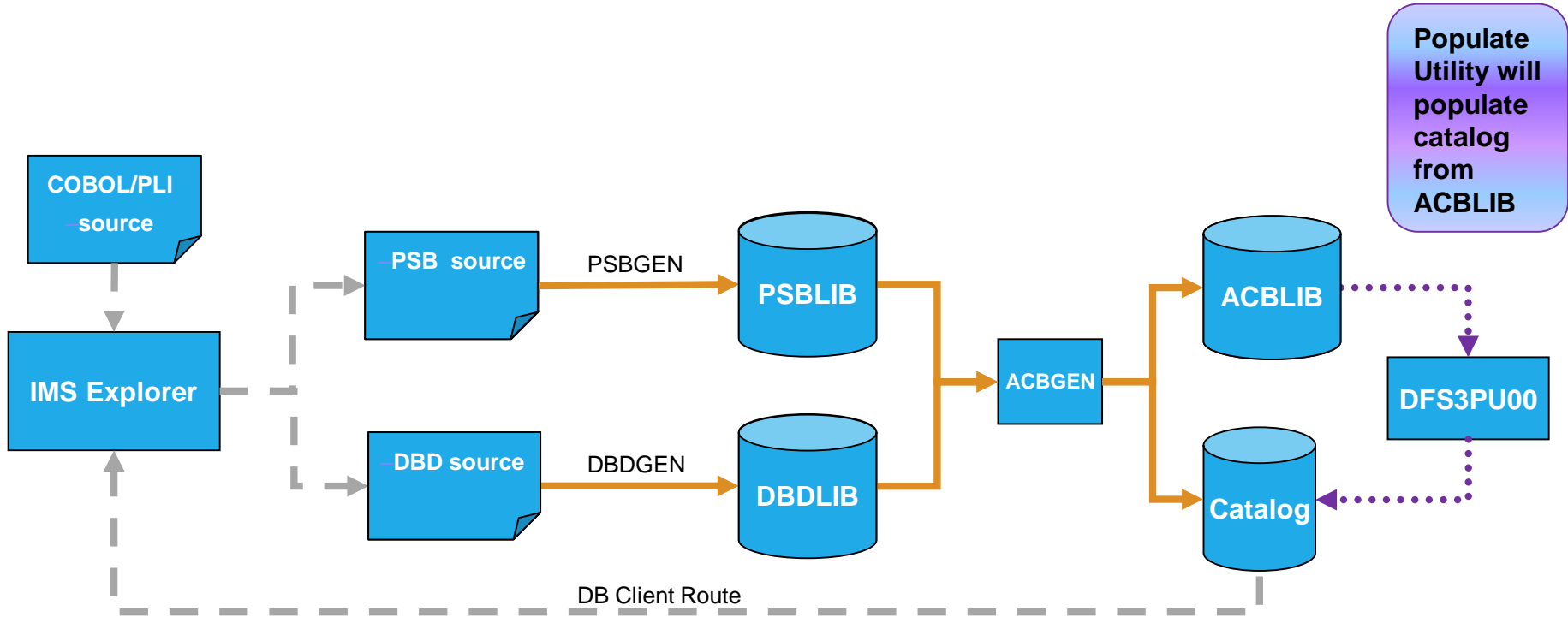


An “instance” of the IMS catalog metadata is **generated each time an ACBGEN is done**. The DFSDFxxx member of the IMS PROCLIB data set is used to configure how many instances are stored and when IMS removes old instances.

What are the Catalog Database dataset names?

- Remember: this is a partitioned database
- For each partition:
 - <HALDB data set prefix>.A00001
 - <HALDB data set prefix>.B00001
 - <HALDB data set prefix>.C00001
 - <HALDB data set prefix>.D00001
 - <HALDB data set prefix>.X00001 ← HIDAM Index
 - <HALDB data set prefix>.L00001 ← ILDS data set

How does metadata get into the catalog?

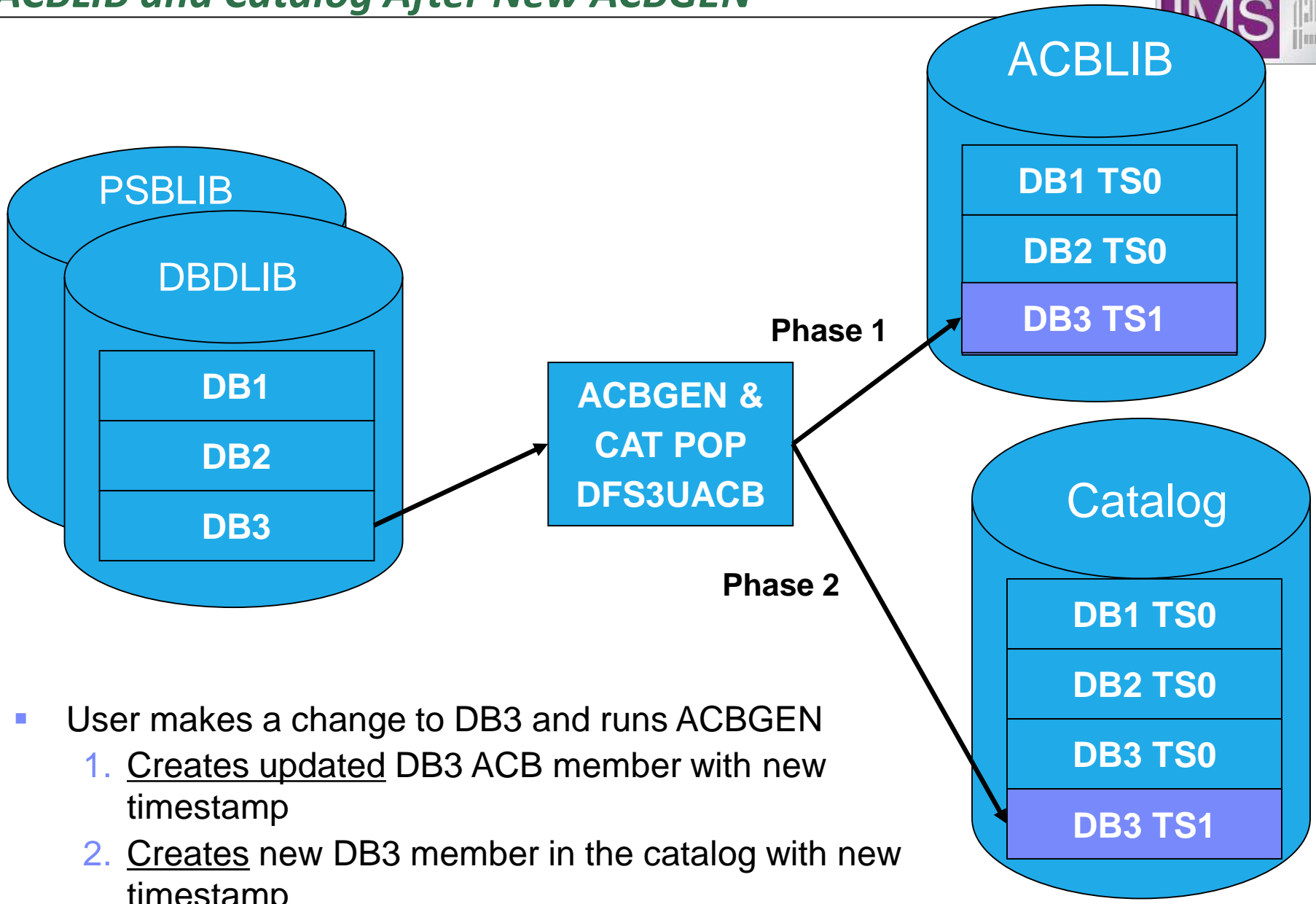
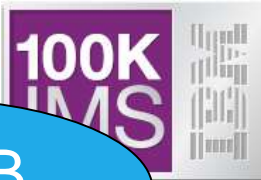


- ACBGEN will populate ACBLIB and catalog in same UOW
 - Populates ACBLIB with *standard* ACB info and *extended* info
 - Populates the catalog with *extended* info
- Key points
 - Only way to update catalog is via the Populate Utility or ACBGEN process
 - Extended info is acquired via the IMS Explorer
 - Extended info stored in ACBLIB members for recoverability

- ACB Generation and Catalog Populate utility, DFS3UACB
 - Replaces existing ACBGEN Utility, DFSUACB0, if IMS catalog enabled
 - Generate ACBLIB member and create catalog metadata in a single job
 - Phase 1 - ACBGEN
 - DBDLIB and PSBLIB members used as input
 - Validation is unchanged
 - ACB member is written to ACBLIB with new ACBGEN timestamp
 - Phase 2 – IMS catalog update
 - Generated ACB is decoded, converted to catalog format, loaded into the catalog
 - DBD and PSB metadata created and inserted
 - Corresponding ACB member timestamp saved as timestamp in catalog DBD and PSB segments
 - Ensures validity and consistency of ACBLIB and catalog
- ACBGEN (DFS3UACB), Catalog Populate (DFS3PU00) and Purge utility are the **only updaters of the IMS catalog**
 - IMS online and IMS batch regions will never update catalog data
 - IMS online and IMS batch regions will only retrieve data from the catalog

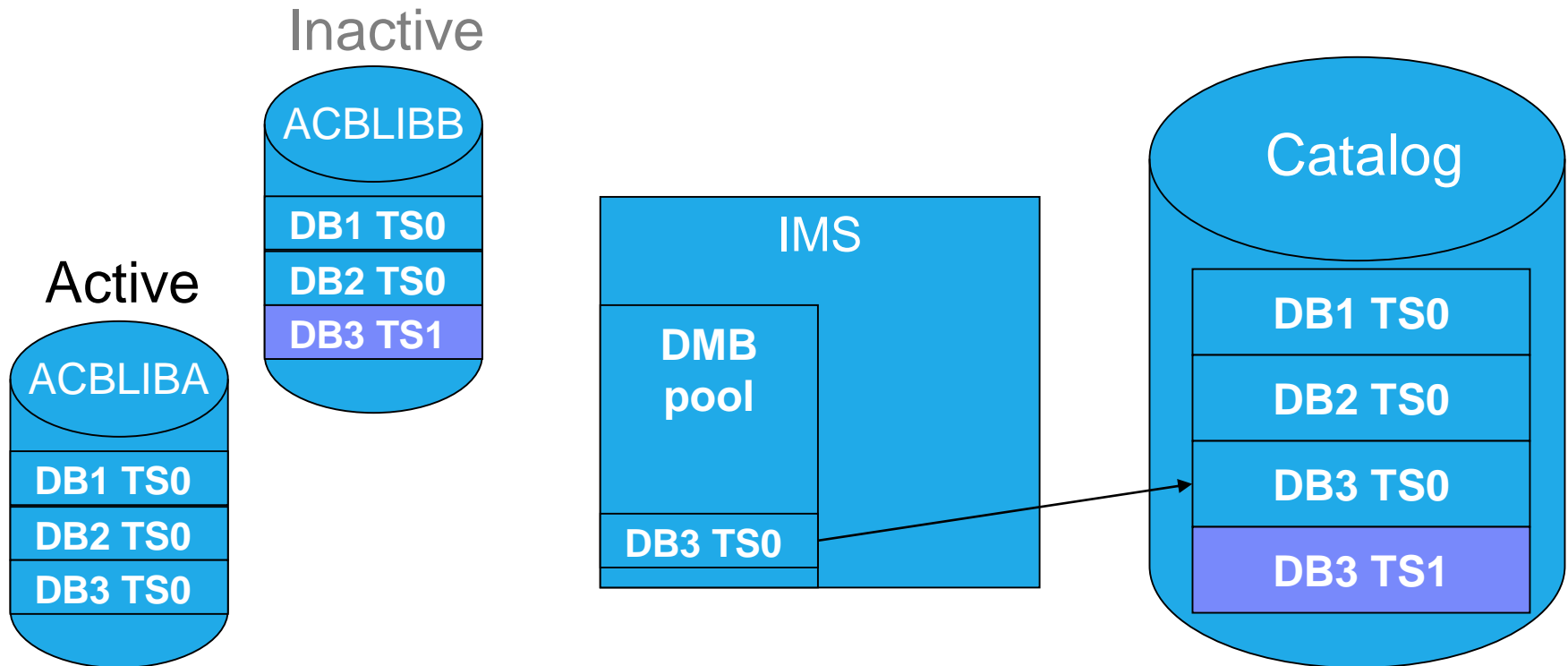
Understanding Catalog member instances

ACBLIB and Catalog After New ACBGEN

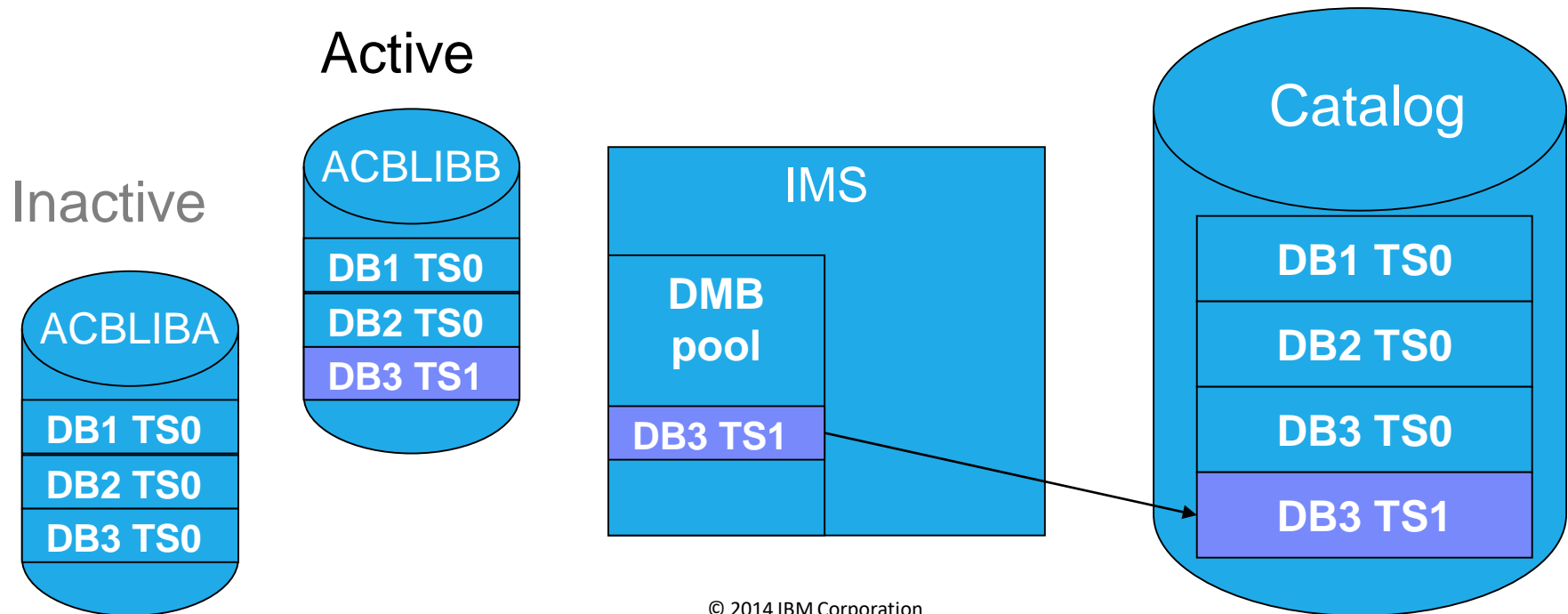


- User makes a change to DB3 and runs ACBGEN
 1. Creates updated DB3 ACB member with new timestamp
 2. Creates new DB3 member in the catalog with new timestamp

- Application request is made to read DB3
 - IMS determines active DB3 member has timestamp TS0
 - Internal DL/I call issued to retrieve member DB3 from IMS catalog
 - IMS retrieves catalog member DB3 with timestamp TS0

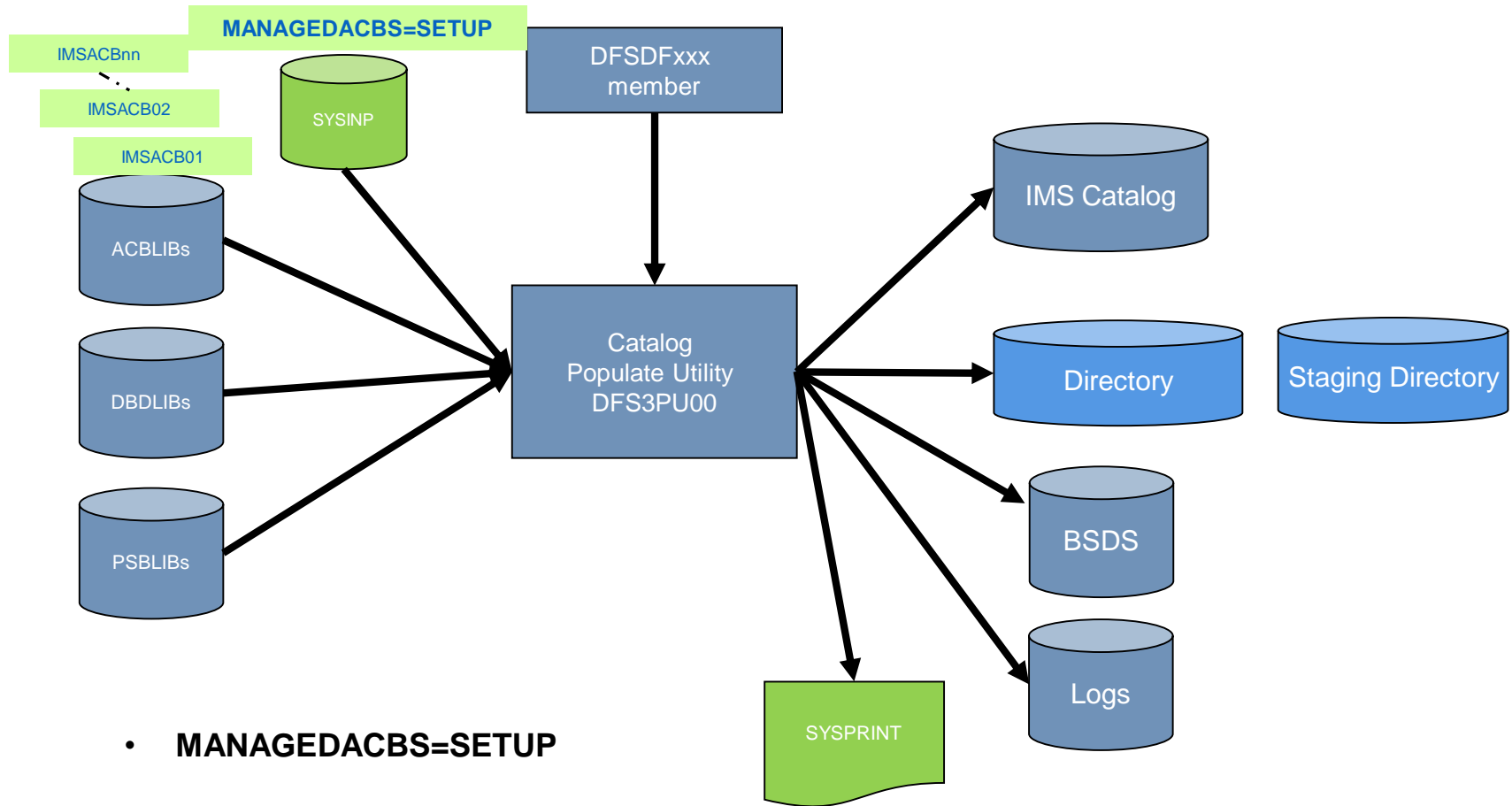


- Initiate OLC to switch from ACBLIBA to ACBLIBB
 - Activates DB3 ACB with timestamp TS1
- Application request is made to read DB3
 - Internal DL/I call issued to retrieve member DB3 from IMS catalog
 - IMS determines active DB3 member has timestamp TS1
 - IMS retrieves catalog member DB3 with timestamp TS1



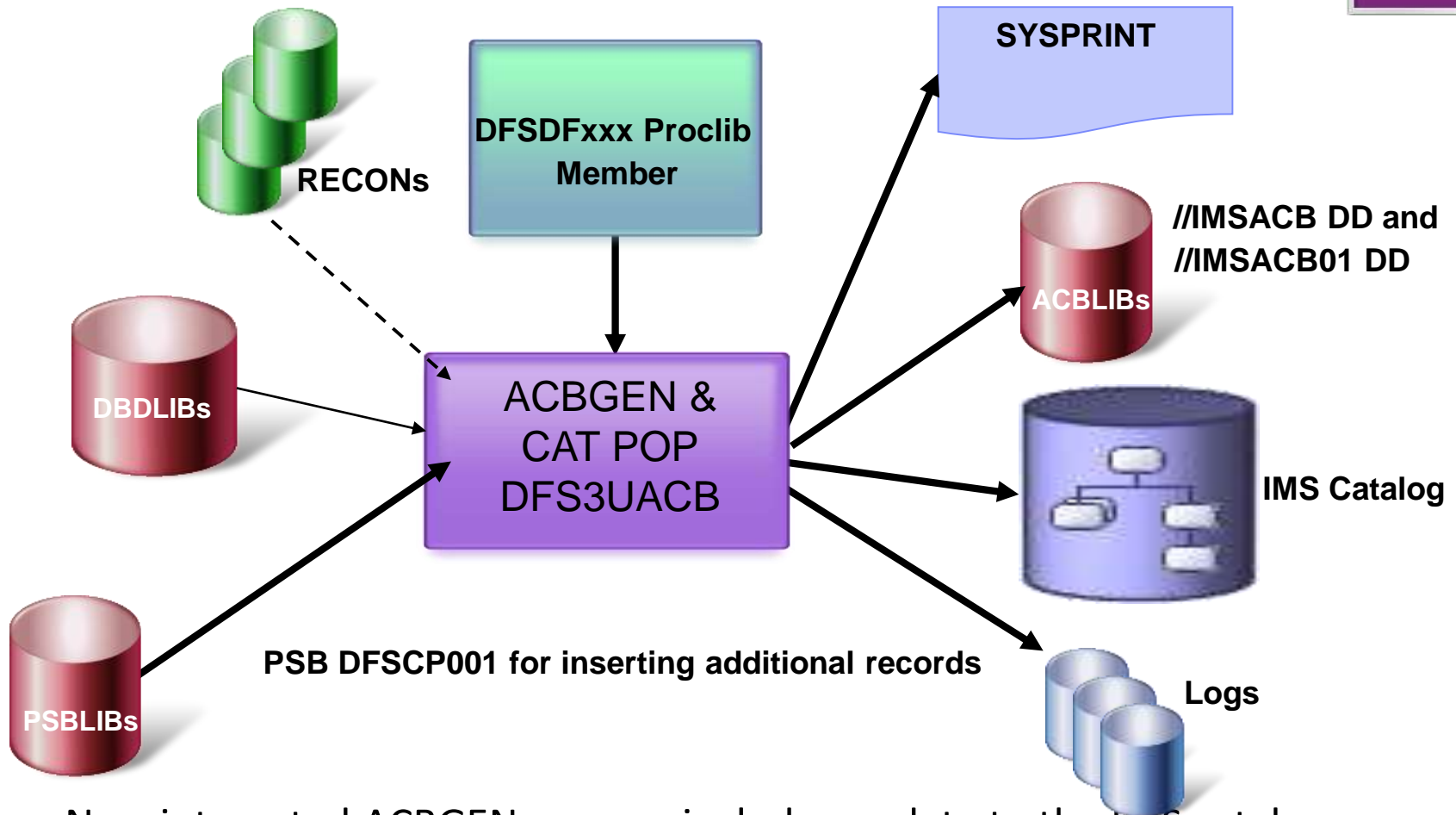
- New “GUR” DL/I call
 - Get Unique Record
 - Restricted to use with IMS Catalog database
- Functions like a GU followed by a series of GNP calls
- Returns the entire database record in one call
 - Saves overhead of issuing GU & GNP to retrieve all the metadata for a catalog member
 - Using an AIB token, the call can be continued if the I/O area is too small for entire catalog database record
- Data returned will be in XML format
 - The XML schemas are included in the IMS.ADFSSMPL data set:
 - DFS3XDBD.xsd (for DBD records)
 - DFS3XPSB.xsd (for PSB records)

So who are the updaters of the Catalog?



- **MANAGEDACBS=SETUP**
- **LOAD PSB DFSCPL00**

IMS Catalog Additions Via ACBGEN & Populate Utility



- New integrated ACBGEN process includes update to the IMS catalog
- DFSDFxxx PROCLIB member has the catalog information
- DFSMDA member used to dynamically allocate the catalog datasets (if not using DBRC for the Catalog)

IMS Catalog Record Purge Utility - DFS3PU10

- IMS Catalog Record Purge utility can be used to remove older instances of metadata from the IMS Catalog
- DFSDFxxx PROCLIB member provides defaults for the utility
- The utility can now run online

IMS Catalog Implementations / Configurations



- Catalog is not Implemented

- Catalog is implemented but IMS Managed ACBs is not
 - i.e. IMS is still using ACBLIBs
 - We have some clients that do this before implementing IMS Managed ACBs

- IMS Managed ACBs is implemented
 - i.e Catalog & Directory are implemented. So IMS is using Directory & not ACBLIBs

*IMS Catalog
Implementations in a
non-IMS Managed ACB
Environment*



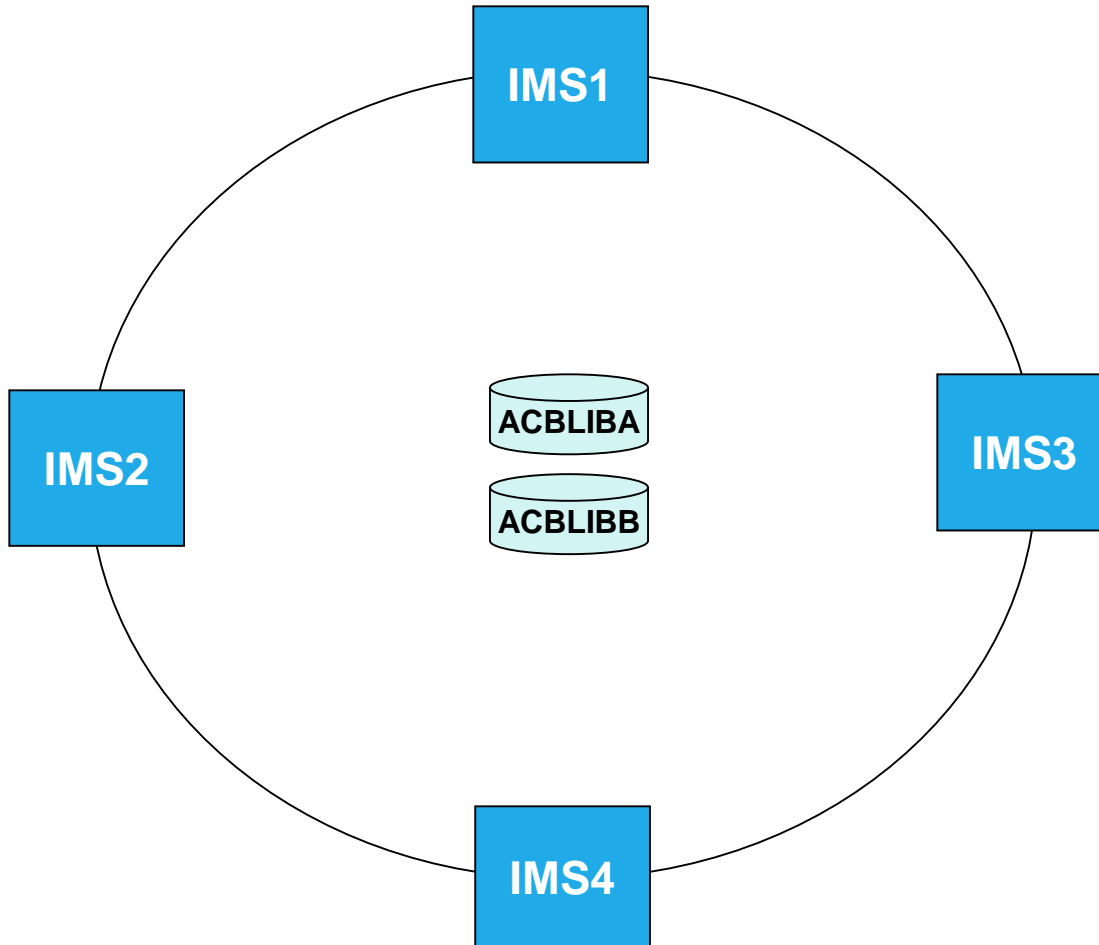


- The IMS catalog supports many environments:
 - one-catalog-per-system,
 - one-catalog-per-sysplex, or
 - any combination of shared and independent catalogs.

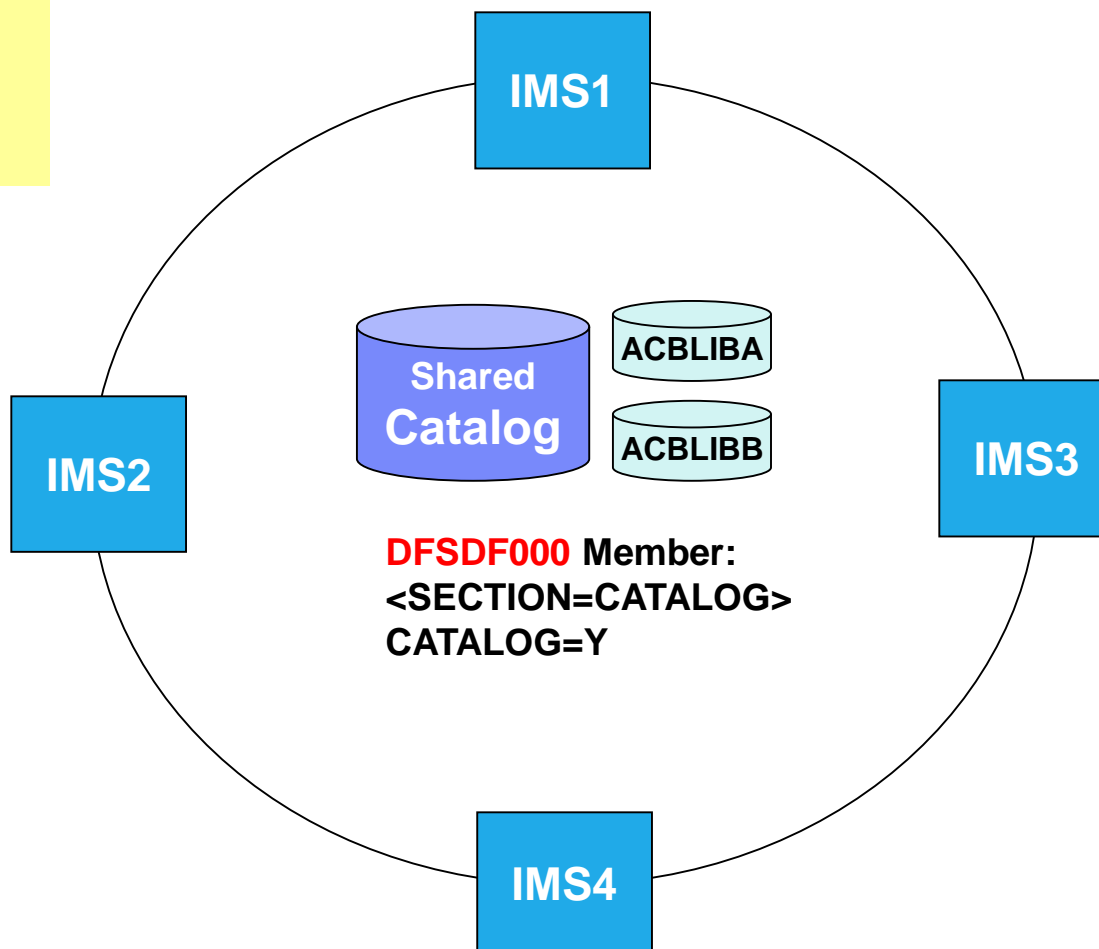
- Each IMS system can be linked to only one instance of the IMS catalog, regardless of system configuration.

Without an IMS Catalog

Multiple IMSes, shared ACBLIBs

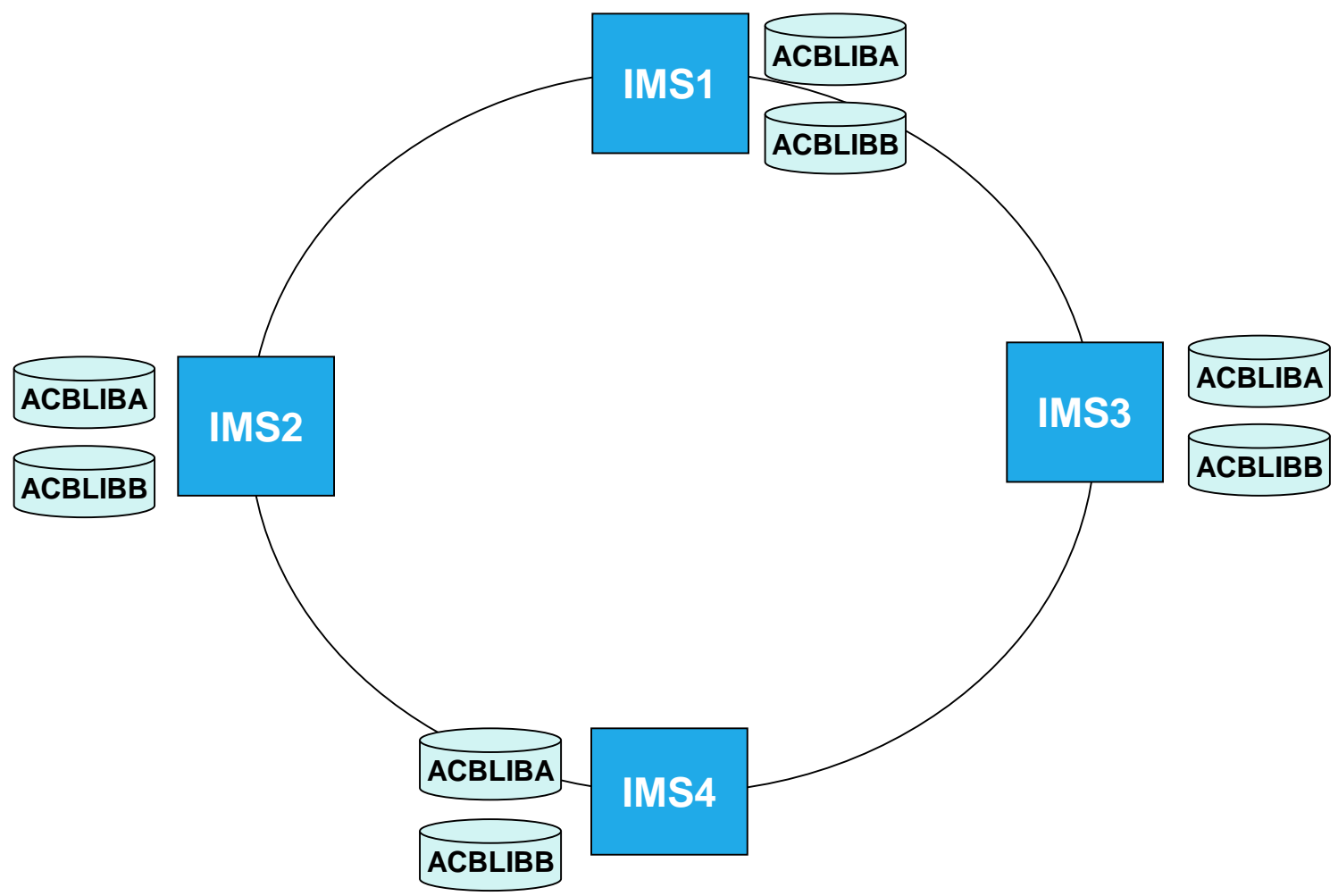


Shared CATALOG DB
Shared DFSDFxxx



Without an IMS Catalog

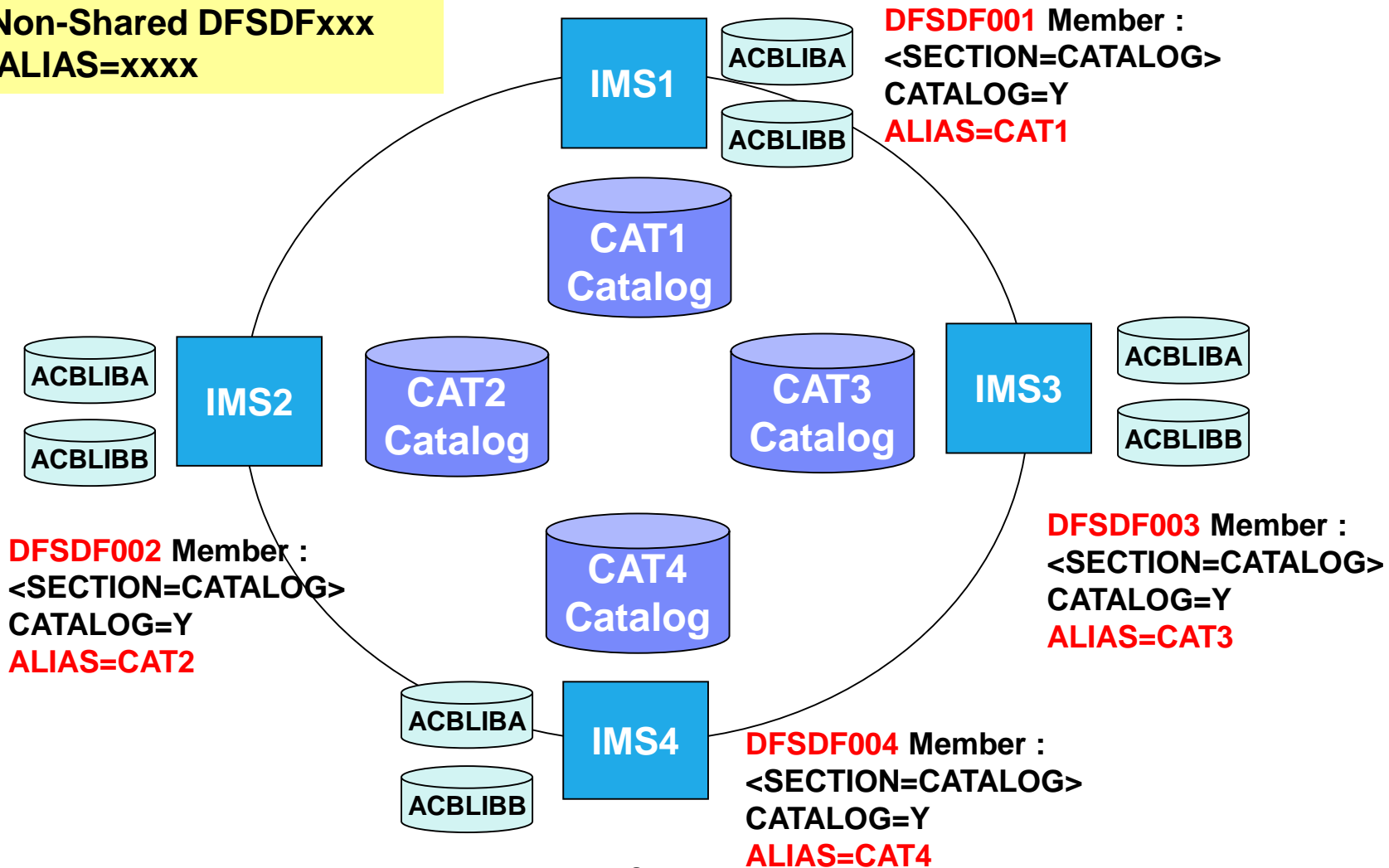
Multiple IMSes, each IMS has it's own cloned ACBLIBs



Multiple IMSes, Cloned ACBLIBs, each IMS has its own Catalog



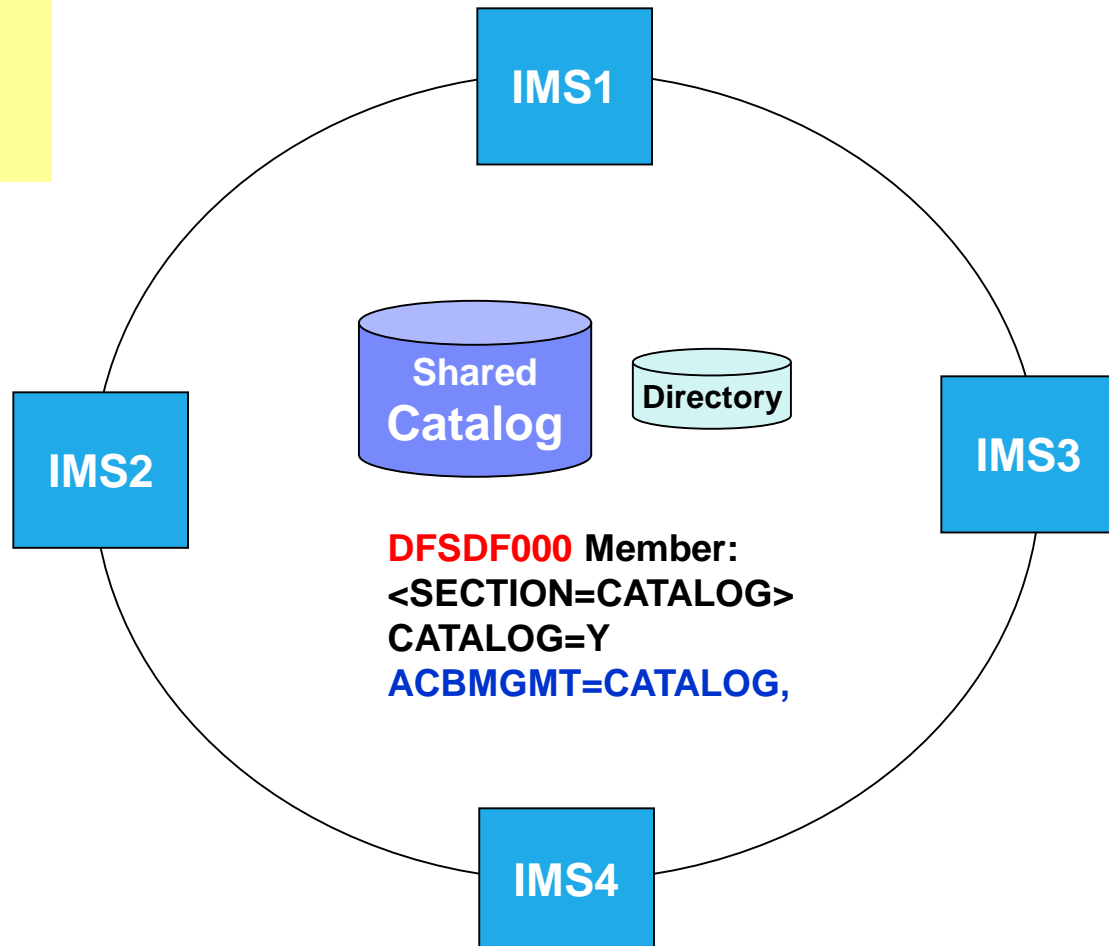
Non-Shared CATALOG DB
Non-Shared DFSDFxxx
ALIAS=xxxx



*IMS Catalog
Implementations in an
IMS Managed ACB
Environment*



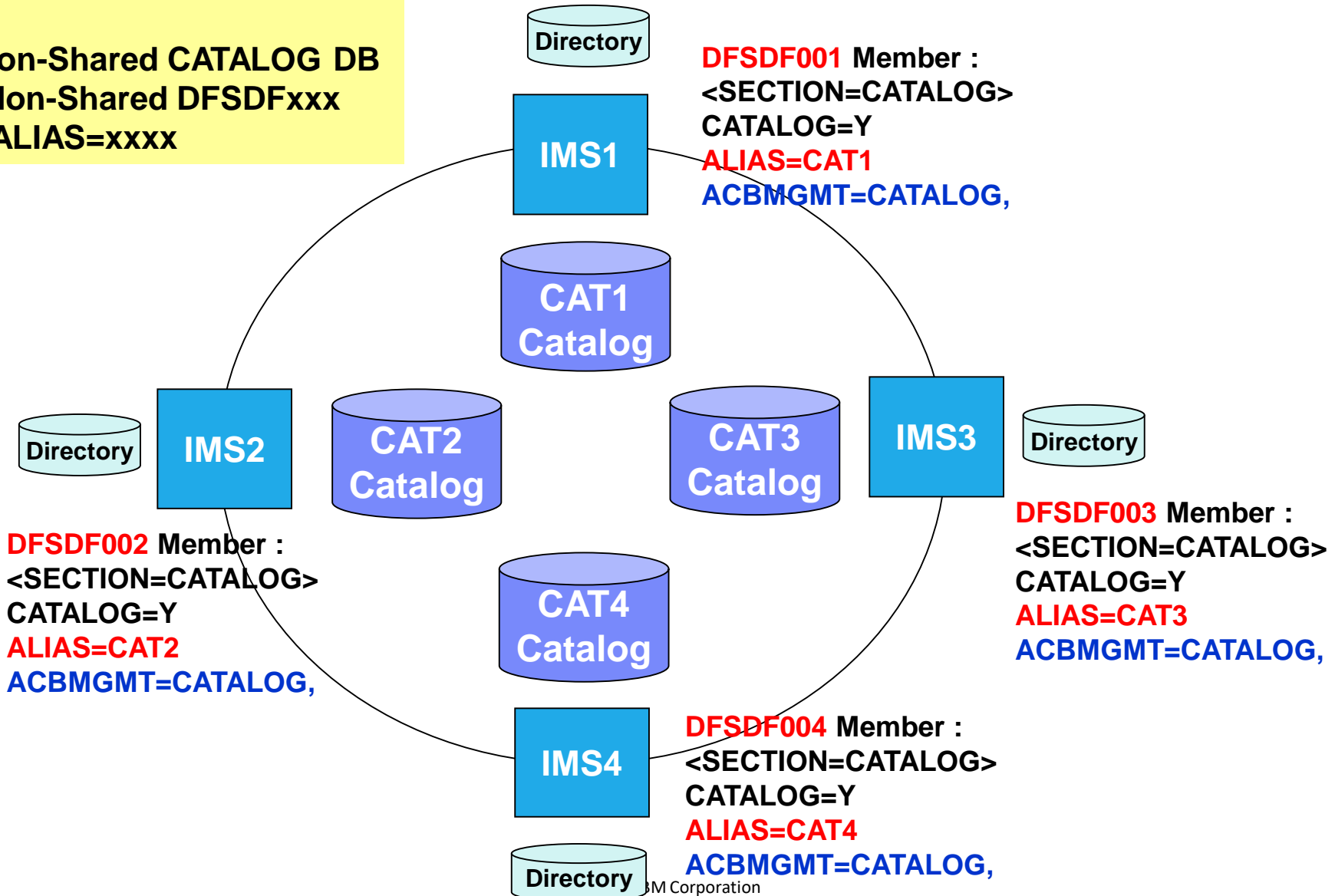
Shared CATALOG DB
Shared DFSDFxxx



Multiple IMSes, Each IMS has its own Catalog & Directory



Non-Shared CATALOG DB
Non-Shared DFSDFxxx
ALIAS=xxxx



IMS Catalog Enablement



Steps to Enable the IMS Catalog



- 1 ■ Copy the IMS supplied DBDs and PSBs for the IMS catalog from the IMS.SDFSRESL data set to your IMS.DBDLIB and IMS.PSBLIB data sets.
- 2 ■ Run the ACB Maintenance utility to generate the ACBs for the IMS catalog.
- 3 ■ Activate the ACB library that contains the IMS catalog ACB.
- 4 ■ Define the HALDB master and partitions of the IMS catalog.
- 5 ■ Code the CATALOG section of the DFSDFxxx member in the IMS.PROCLIB data set.
- 6 ■ If you need to manually allocate the database data sets for the IMS catalog, allocate the data sets now. Otherwise, IMS creates them automatically.
- 7 ■ Load the IMS catalog by running the DFS3PU00 utility.
- 8 ■ Image copy the Catalog
- 9 ■ Restart IMS



- 1 Add IMS supplied catalog DBDs and PSBs into your DBDLIB and PSBLIB
 - Copy DBD and PSB object code from SDFSRESL to your DBDLIB and PSBLIB

```
//CPYCMEM EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SDFSRESL DD DSN=IMSCFG.IMB1.SDFSRESL,DISP=SHR
//DBDLIB DD DSN=IMSCFG.IMSB.DBDLIB,DISP=OLD
//PSBLIB DD DSN=IMSCFG.IMSB.PSBLIB,DISP=OLD
//SYSIN DD *
        COPY OUTDD=DBDLIB,INDD=( (SDFSRESL,R) ),LIST=YES
        SELECT MEMBER=(DFSCD000,DFSCX000)
        COPY OUTDD=PSBLIB,INDD=( (SDFSRESL,R) ),LIST=YES
        SELECT MEMBER=(DFSCPL00,DFSCP000,DFSCP001,DFSCP002,DFSCP003)
```




N.B. The MODBLKS resources for the Catalog databases and Programs do not need to be defined!



2 ■ Run an ACB Gen for the Catalog PSBs

```
//PROCS JCLLIB ORDER=(IMSCFG.IMSB.PROCLIB)
//ACBGEN EXEC PROC=ACBGEN,SOUT='*',COMP='POSTCOMP'
//G.SYSIN DD *
    BUILD PSB=(DFSCPL00)
    BUILD PSB=(DFSCP000)
    BUILD PSB=(DFSCP001)
    BUILD PSB=(DFSCP002)
    BUILD PSB=(DFSCP003)
//*
```



ACBGEN can be run into Staging Library

3 ■ Activate the ACB library that contains the IMS Catalog ACBs

- Run the Online Change Copy Utility to “Backup” Active ACBLIB in Inactive ACBLIB
- Initiate an ACB Member Online Change

```
INIT OLC PHASE(PREPARE) TYPE(ACBMBR) NAME(DFSCD000,DFSCX000)
INIT OLC PHASE(COMMIT)
```

- Run the Online Change Copy Utility to Copy Active ACBLIB in Inactive ACBLIB

4

Definition of the HALDB structure

- Partitioning of the catalog is the users responsibility
 - Minimum of 1 partition is required
 - Last partition must be able to contain the highest-key PSB record
 - Catalog HALDB uses the high-key selection method
 - No use of Partition Selection Exit is allowed
- Catalog Database Definition
 - For systems that use DBRC
 - Catalog database can be defined to the RECONS with the DBRC utility and commands
 - For systems that do not use DBRC
 - Catalog database must be defined to the Catalog Partition Definition data set using the Catalog Partition Definition Data Set utility, DFS3UCD0
 - If an ALIAS is used in the CATALOG sections of the DFSDfxxx member, each alias Catalog database must be defined


- DBRC DSPURX00 utility and commands to define the Catalog DB to the RECONS

```
//D.SYSIN DD *
  INIT.DB      DBD(DFSCD000) TYPHALDB SHARELVL(3)
  INIT.PART    DBD(DFSCD000) PART(DFSCD01) -
              DSNPREFIX(IMSCFG.IMSC.DFSCD000) -
              BLOCKSIZE(4096,4096,4096,4096) -
              KEYSTRNG(X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF') -
              ICJCL(ICJCL) OICJCL(OICJCL) RECOVJCL(RECOVJCL) -
              NOREUSE RECOVPD(0) GENMAX(3) RECVJCL(RECVJCL)
  INIT.DB      DBD(DFSCX000) TYPHALDB SHARELVL(3)
  INIT.PART    DBD(DFSCX000) PART(DFSCX01) -
              DSNPREFIX(IMSCFG.IMSC.DFSCX000) -
              KEYSTRNG(X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF') -
              ICJCL(ICJCL) OICJCL(OICJCL) RECOVJCL(RECOVJCL) -
              NOREUSE RECOVPD(0) GENMAX(3) RECVJCL(RECVJCL)
```

- If using a catalog alias prefix, replace DFSC in the database and partition names for the catalog and the catalog secondary index with the four character ALIAS name prefix
- You might need to define multiple alias name databases to the RECONS

- Using the Catalog Partition Definition Data Set utility, DFS3UCD0, to define the Catalog database (for systems that do not use DBRC)

```
//S1 EXEC PGM=DFS3UCD0,REGION=0M
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSHDBSC DD DSN=...,DISP=
//SYSPRINT DD SYSOUT=*
//IMS DD DSN=IMS.DBDLIB,DISP=SHR
//SYSIN DD *
HALDB=(NAME=DFSCD000)
PART=(NAME=DFSCD000,PART=partitionname,
      DSNPREFIX=dsnprefix,
      KEYSTRNG=keystring)
HALDB=(NAME=DFSCX000)
PART=(NAME=DFSCX000,PART=partitionname,
      DSNPREFIX=dsnprefix,
      KEYSTHEX=FFFFFFFFFFFFFFFF) /*
```



DFSHDBSC is the DDNAME for the Catalog Partition definition data set. IVP provides sample jobs and tasks that demonstrate how to set-up a simple IMS Catalog.

**Unregistered
Catalog database**

- Catalog Partition Definition data set is populated with the information specified in the HALDB and PART control cards
 - RECON-like information for catalog database partition definition and structure
- The name DFSCD000 in the HALDB and PART statements contains the default catalog prefix DFSC. If your catalog uses an alias name prefix, substitute it in the JCL

- After Catalog database is defined in Catalog Partition Definition Data Set
 - Identify unregistered Catalog database names
 - **UNREGCATLG** parameter in the DATABASE section of the DFSDfxxx member

```
/* **** */
/* Database Section */
/* **** */
<SECTION=DATABASE>
UNREGCATLG=(DFSCD000,DFSCX000) /* Unregistered IMS catalog DB */
/* **** */
/* **** */
/* **** */
```

- If using an alias name prefix, replace DFSC in the UNREGCATLG database names with the four character alias name prefix
- Limitations of using an unregistered Catalog database
 - NO IMS Data Sharing support
 - NO OLR support
 - NO partition definition change support
 - User must rebuild catalog partitions
 - Manual recovery required for unregistered Catalog databases

**Unregistered
Catalog database**

- After Catalog database is defined in Catalog Partition Definition Data Set
 - Create a new DFSMDA dynamic allocation member for the Catalog Partition Definition data set

```
//DYNALOC JOB
//*
//STEP EXEC IMSDALOC
//SYSIN DD *
    DFSMDA TYPE=INITIAL
    DFSMDA TYPE=CATDBDEF, DSNAME=dsn
    DFSMDA TYPE=FINAL
    END
/*
```

Unregistered
Catalog database

- *dsn* is the name of the Catalog Partition Definition data set
 - Catalog Partition Definition data set was allocated in the DFS3UCD0 utility on the **DFSHDBSC DD**
- Dynamically allocate Catalog Partition Definition data set in any IMS job step

5

Modify DFSDFxxx PROCLIB Member

– New CATALOG section(s) for catalog related parameters

- Single section format <SECTION=CATALOG>
- Multiple section format <SECTION=CATALOG $imsid$ >
 - Multiple IMS systems sharing one DFSDFxxx PROCLIB member
 - $imsid$ suffix must be a four character IMS ID

– CATALOG section parameters

- CATALOG=N | Y
 - Catalog is disabled or enabled
 - If enabled, IMS automatically creates catalog DDIR & PDIRs at IMS startup
- ALIAS=DFSC | xxxx **(no default value)**
 - Specifies any 1-4 alphanumeric value used as a Catalog database name prefix
 - Enables use of non-shared, aliased, Catalog databases within an IMSplex
 - Use in a data sharing environment where each IMS has its own Catalog database and all are registered in a single set of RECONS
 - At runtime, the alias Catalog database names are dynamically replaced with internal database names **DFSCD000** and **DFSCX000**
 - For standalone IMS system use “**DFSC**” which is the standard Catalog database name prefix → **DFSCD000** and **DFSCX000**

The Catalog Definition user exit routine (DFS3CDX0) is provided as an alternative option for batch processing environments that do not use the DFSDFxxx member of the PROCLIB data set.



■ DFSDFxxx PROCLIB Member

– CATALOG section parameters (continued)

- Information used by Catalog Populate Utility to ***automatically allocate*** the Catalog database data sets
 - DATACLAS
 - Optional data class for SMS managed data sets
 - MGMTCLAS
 - Optional management class for SMS managed data sets
 - STORCLAS
 - Required storage class for SMS managed data sets
 - IXVOLSER
 - Volume serial number for primary and secondary catalog indices
 - Required for non-SMS managed data sets - *the secondary index will be created with a CI SIZE of 18432 - so buffers of 20480 are required in the DFSVSMxx proclib member*
 - SPACEALLOC
 - Free space % (0 to 9999) added to the IMS-computed size of the primary & secondary data set allocations
 - SMSVOLCT
 - Number of volumes (1-20) created by the Catalog Populate utility for SMS managed data sets

■ DFSDFxxx PROCLIB Member

– CATALOG section parameters (continued)

- RETENTION=(INSTANCES=nnnnn,DAYS=nnnnn)

- Specifies retention schedule for metadata in the IMS catalog

- By default IMS keeps only two instances of the DBD or PSB in catalog

- Metadata instances older than the specified retention period are not automatically deleted, but available for removal when a new instance of metadata is added

- Default value of “0” disables this feature

- You can specify the INSTANCES parameter, the DAYS parameter, or both.

- GURCACHE= 1

- Specifies the amount of storage, in gigabytes, to allocate in 64-bit memory to cache XML documents generated as responses to GUR calls. The valid values are 1 through 999.

You need to run the Catalog Purge Utility to delete DBDs and PSBs from the Catalog – it doesn't happen automatically.



Defining an Alias for the IMS Catalog



- Use in a data sharing environment where each IMS has its own Catalog database and all are registered in a single set of RECONS
- At runtime, the alias Catalog database names are dynamically replaced with internal database names **DFSCD000** and **DFSCX000**
- Steps:
 1. In DFSDFxxx Proclib member (in Catalog section) code ALIAS= 1-4 alphanumeric value used as a Catalog database name prefix.
 2. Add alias names to the catalog database name list in your IMS DBD library with **the IMS Catalog Alias Names utility (DFS3ALIO)**. Modify the following JCL for your environment:

```
//ALIAS EXEC PGM=DFS3ALIO
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DSN=IMS.DBDLIB,DISP=OLD
//SYSLIN DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN DD *
prefix1,prefix2,...
/*
```

Defining an Alias for the IMS Catalog (contd.)



- Define the new catalog database in the RECON data set. Modify the following JCL for your environment. This example uses the alias name IMS1:

```
//ALIAS1 EXEC PGM=DSPURX00
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=*
//IMS DD DSN=IMS.DBDLIB,DISP=SHR
//SYSIN DD *
INIT.DB DBD(IMS1D000) TYPHALDB SHARELVL(3)
INIT.PART DBD(IMS1D000) PART(IMS1D01) -
  DSNPREFIX(dsnprefix.IMS1D000) -
  BLOCKSIZE(4096) -
  KEYSTRNG(X'FFFFFFFFFFFFFFFFFFFFFFFF')
INIT.DB DBD(IMS1X000) TYPHALDB SHARELVL(3)
INIT.PART DBD(IMS1X000) PART(IMS1X01) -
  DSNPREFIX(dsnprefix.IMS1X000) -
  KEYSTRNG(X'FFFFFFFFFFFFFFFFFFFFFFFF')
/*
```




6

If you want to manually allocate the database data sets for the IMS catalog, allocate the data sets now. Otherwise, IMS creates them automatically.

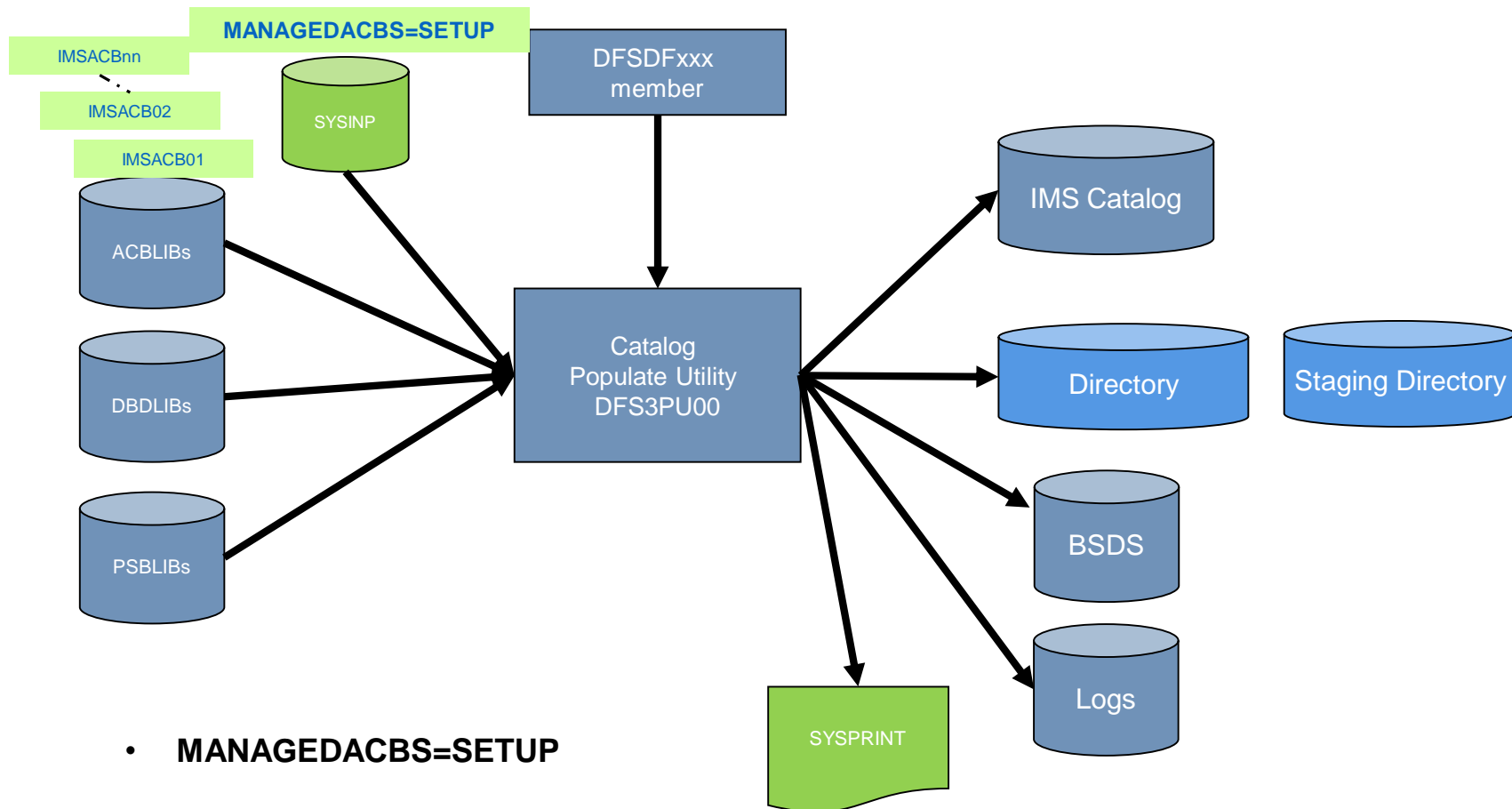
7 Populate the IMS Catalog

- Load the IMS Catalog using IMS Catalog Populate utility, DFS3PU00
 - Each ACB member is decoded, converted to catalog format, loaded into the catalog
- Reads ACBLIB, DBDLIB and PSBLIB datasets as input
 - Data sets can be concatenated but only first occurrence of an ACB member is used
 - DBDLIB needed for Logical databases and GSAM databases
 - PSBLIB needed to determine which GSAM databases go into the catalog
- Catalog database DBD and PSB segments will have a version and contain a timestamp that matches the ACB member timestamp
 - Used to associate an ACB member with a catalog member
 - Timestamp exceptions
 - DBDGEN timestamp for Logical and GSAM DBs
 - PSBGEN timestamp for GSAM only PSBs

■ Populate the IMS Catalog ...

- If any of the database data sets do not exist, the DFS3PU00 utility creates them automatically: i.e. the DFSCD000 database data sets which are:
 - The primary index data set
 - The indirect list data set (ILDS)
 - Four data set group data sets for the segments of the IMS catalog
 - DFSCX000 secondary index data set
- DFS3PU00 calculates the size of the database data sets based on both the size of the ACB library and the expansion percentages that you can specify in the DFSDFxxx PROCLIB member
 - The DFSDFxxx member is also where you specify the volume serial number for the VSAM key-sequenced data sets or the SMS storage class, data class, and management class for all data sets

- IMS Catalog Populate utility
 - Can run as a typical IMS Batch or BMP job
 - Requires IMS logs for backout / recovery
 - Requires IRLM if catalog is shared and catalog active in an IMS subsystem
 - Business as usual for data sharing
 - Requires DBRC if catalog is defined in the RECON
 - If using Catalog Partition Definition Data Set
 - User's responsible to ensure online catalog access has ceased
 - Business as usual for non-registered full function DB

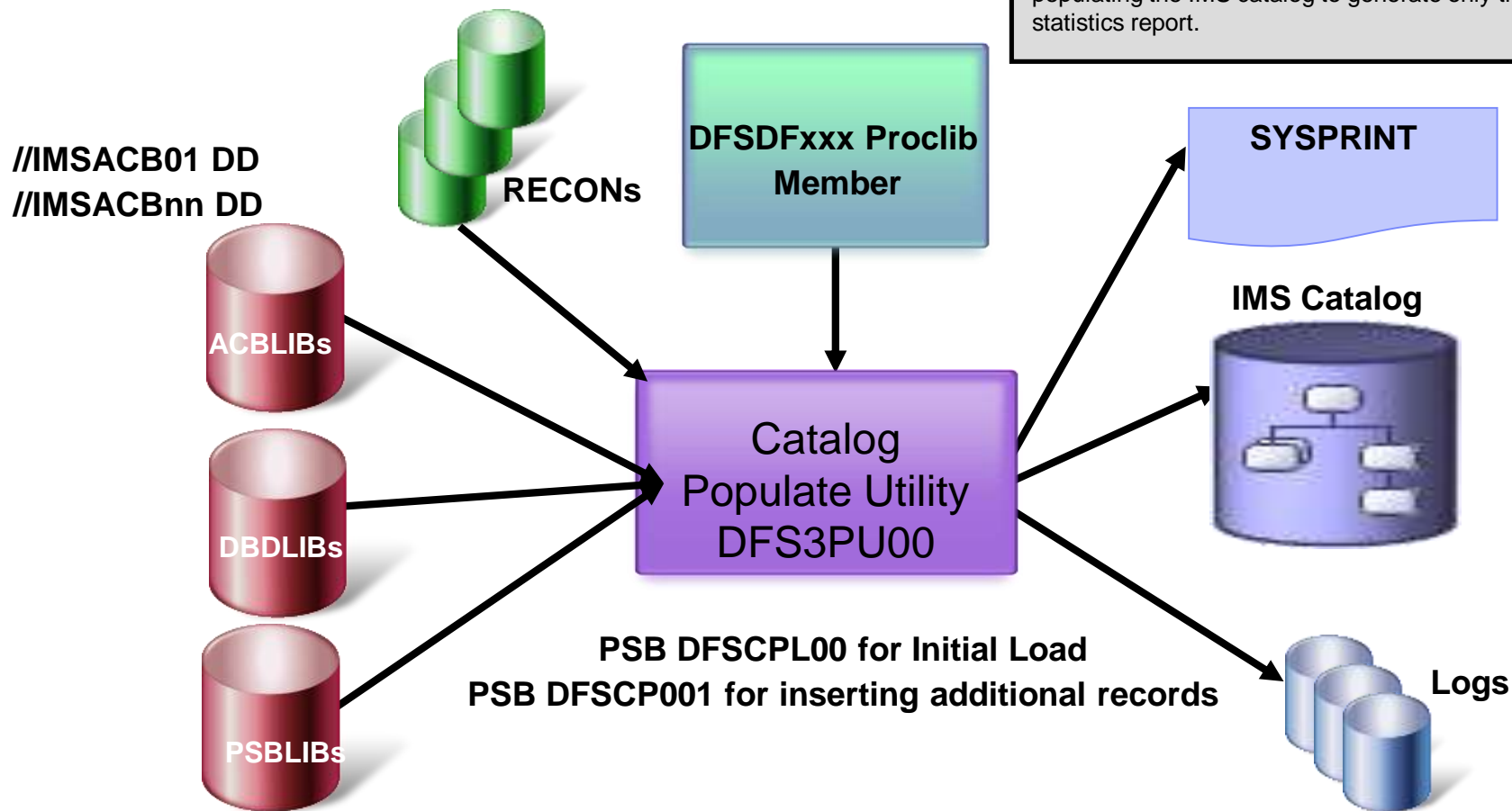


- **MANAGEDACBS=SETUP**
- **LOAD PSB DFSCPL00**

Catalog Populate Utility – DFS3PU00



If you need to know how much DASD storage the IMS catalog data sets will use before they are created, you can run the DFS3PU00 utility without populating the IMS catalog to generate only the statistics report.



Take an Image Copy of the Catalog database after the initial populate!
(Required if using DBRC)

8

- If the Catalog is registered to DBRC

- Catalog database datasets are now marked “Image Copy Needed” in the RECON
- Image Copy the databases

```
DSP0132I  IMAGE COPY NEEDED FOR
DSP0132I  DBDNAME=DFSCD01    DDNAME=DFSCD01A
DSP0132I  IMAGE COPY NEEDED FOR
DSP0132I  DBDNAME=DFSCD01    DDNAME=DFSCD01B
DSP0132I  IMAGE COPY NEEDED FOR
DSP0132I  DBDNAME=DFSCD01    DDNAME=DFSCD01C
DSP0132I  IMAGE COPY NEEDED FOR
DSP0132I  DBDNAME=DFSCD01    DDNAME=DFSCD01D
```

- 9 Restart IMS causing DFSDFxxx to be read and Catalog to be enabled!

```
Response for: QRY DB NAME (DFSC*) SHOW (ALL) More:
DBName      PartName  MbrName      CC TYPE      Acc Status  LAcc LRsdnt  LclStat
DFSC*                IMB2          0              NONE
DFSCD000                IMB1          0 PHIDAM                READ Y
DFSCD000 DFSCD01    IMB1          0 PART                READ      NOTOPEN
DFSCD000 DFSCD01    IMB1          0 PART                READ Y    NOTOPEN
DFSCD000                IMB2          0 PHIDAM                READ Y
DFSCD000 DFSCD01    IMB2          0 PART                READ Y    NOTOPEN
DFSCD000 DFSCD01    IMB2          0 PART                READ      NOTOPEN
DFSCX000                IMB1          0 PSINDEX             READ Y
DFSCX000 DFSCX01    IMB1          0 PART                READ      NOTOPEN
DFSCX000 DFSCX01    IMB1          0 PART                READ Y    NOTOPEN
DFSCX000                IMB2          0 PSINDEX             READ Y
DFSCX000 DFSCX01    IMB2          0 PART                READ Y    NOTOPEN
DFSCX000 DFSCX01    IMB2          0 PART                READ      NOTOPEN
```


- Catalog database management is required
 - Review/adjust database buffer pool definitions
 - Perform routine management and maintenance on the Catalog database
 - Image Copy, Pointer Checker, Reorg, etc...
 - Catalog database will need to be REORG'd
 - If Catalog database is defined to the RECONS →
 - HALDB OLR non-disruptive reorganization is supported
 - If Catalog database is not defined to the RECONS →
 - HALDB OLR can't be supported and a reorg utility must be employed
- Support with existing backup and recovery procedures
 - Image copy, recovery, backout utilities etc...
 - If Catalog database is not defined to the RECONS → recovery is limited
 - Same as non-registered, full function database recovery procedure

- Run the Catalog Purge Utility DFS3PU10 to remove Catalog entries

```
//BATCH EXEC PGM=DFSRR00,  
// PARM=(DLI,DFS3PU10,DFSCP001,,,,,,,,,IMB1,,Y,Y,IRLB,,,,,,,,,  
//      'DFSDF=0B1')  
//STEPLIB DD DSN=IMSCFG.IMB1.SDFSRESL,DISP=SHR  
//      DD DSN=IMSCFG.IMSB.USERLIB,DISP=SHR  
//DFSRESLB DD DSN=IMSCFG.IMB1.SDFSRESL,DISP=SHR  
//PROCLIB DD DSN=IMSCFG.IMSB.PROCLIB,DISP=SHR  
//IMS      DD DSN=IMSCFG.IMSB.PSBLIB,DISP=SHR  
//      DD DSN=IMSCFG.IMSB.DBDLIB,DISP=SHR  
//SYSUT1 DD DSN=IMSCFG.IMSB.CATPURG,DISP=OLD  
//IEFRDER DD DSN=&&IMSLOG1,DISP=(NEW,DELETE),  
// DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096,BUFNO=5),  
// SPACE=(CYL,(200,75)),UNIT=SYSDA  
//IEFRDER2 DD DSN=&&IMSLOG2,DISP=(NEW,DELETE),  
// SPACE=(CYL,(200,75)),UNIT=SYSDA  
//DFSVSAMP DD DSN=IMSCFG.IMSB.PROCLIB(DFSVSMB),DISP=SHR  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
MODE BOTH  
//*
```



- IMS Catalog Overview
- IMS Catalog Configurations
- IMS Catalog Enablement