bmcsoftware

ACTIVATE BUSINESS WITH THE POWER OF I.T.™

DB2 Directory – What, How, When, and Where

Bill Arledge, Consulting Product Manager
BMC Software Inc.

12/12/2008

Overview

- Catalog & Directory – Introductory Details
- DB2 Directory Details
- Operational Considerations
- Utility Considerations

DB2 System Databases

DB2 CATALOG
DSNDB06

- Metadata for DB2 objects defined in the subsystem
- Updated by DB2 in response to certain SQL statements, command, and utilities
- Accessible using standard SQL

DB2 DIRECTORY
DSNDB01

- Operational run-time components
- Populated by DDL, bind operations, and some real-time events related to recovery
- Not visible in any catalog views

DB2 WORKFILE
User defined

- Work files used for various purposes
  - Global temporary tables
  - SQL work space (result sorting, view materialization)
- DB2 9 merged Workfile and Temp into Workfile
- Can be expanded by adding additional tablespaces

Default DATABASE
DSNDB04

- Database for tablespaces defined with no IN DATABASE specification

Default DATABASE
DSN*nnnnn*

- Implicitly defined databases supported in DB2 9

DB2 System Databases
"Catalog" Tablespaces & Tables

28 Tablespaces
104 Tables

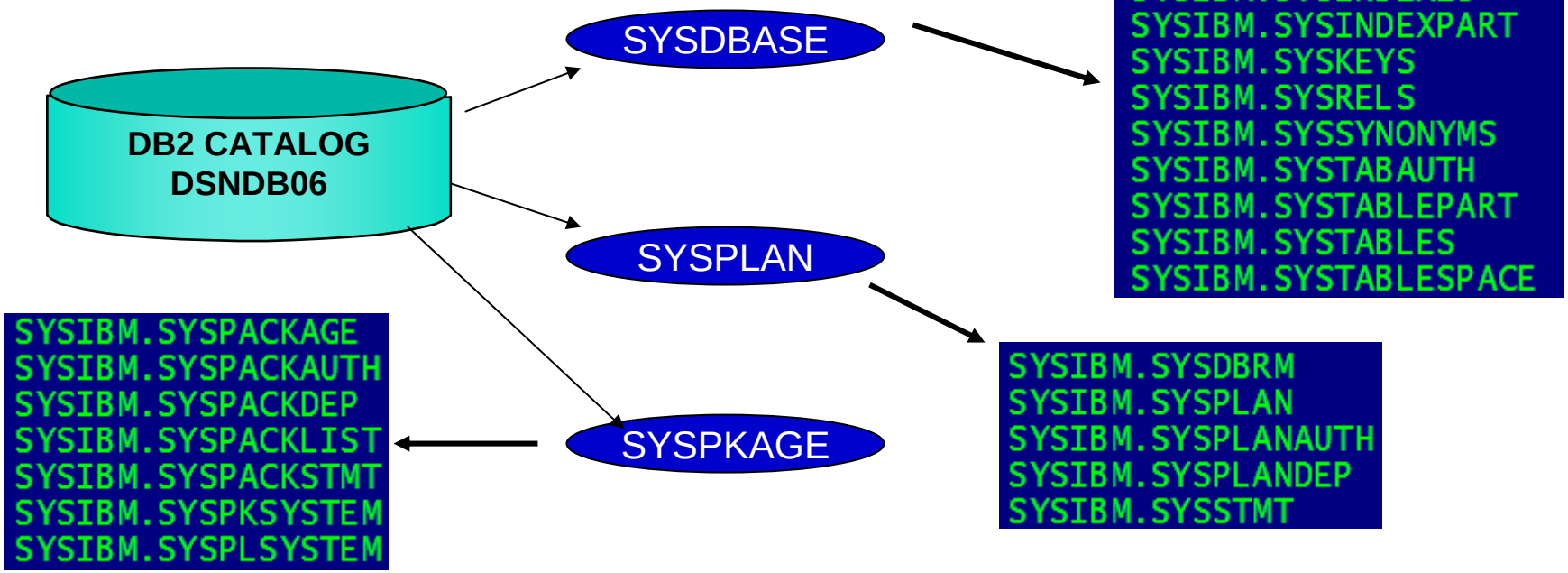
DB2 CATALOG
DSNDB06

- SYSALTER
- SYSXML
- SYSCONTX
- SYSVIEWS
- SYSCOPY
- SYSUSER
- SYSDBASE
- SYSTARG
- SYSDBAUT
- SYSSTR
- SYSDDF
- SYSSTATS
- SYSEBCDC
- SYSSEQ2
- SYSGPAUT
- SYSSEQ
- SYSGROUP
- SYSRTSTS
- SYSGRTNS
- SYSROLES
- SYSHIST
- SYSPLUXA
- SYSJAUXA
- SYSPLAN
- SYSJAUXB
- SYSPKAGE
- SYSJAVA
- SYSOBJ

- Object definitions
- Authorizations
- Plans & Packages
- DB2 Statistics
- Communications definitions
- And other stuff

DB2 System Databases
A Bit More on Catalog Tables

- Metadata for all DB2 objects
- Populated/updated in response to certain SQL, commands, and utilities
- Accessible via SQL DML statements
- Examples of a few of the more important catalog tables
  - These particular tables are related to the contents of the DB2 Directory



DB2 System Databases Directory Structures

DBD01

Database Descriptors

- A run-time operational structure that identifies all DB2 objects created in a specific database
- Internal structure not visible to the “naked” eye

SCT02

Skeleton Cursor Table

- Operational run-time structure with access path information about a DBRM-based plan

SPT01

SPT01 – Skeleton Package Table

- Operational run-time structure with access path information about packages

SYSLGRNX

SYSLGRNX – Recovery Log Ranges

- Provides details by DB2 object about log ranges required for recovery processing

SYSUTILX

DB2 In-flight Utilities Register

- Contains entries about DB2 utilities currently running or abnormally terminated/stopped

DB2 System Databases
The “Directory” Tablespaces

Database Descriptors

DBD01

Skeleton Cursor Tables

SCT02

Skeleton Package Tables

SPT01

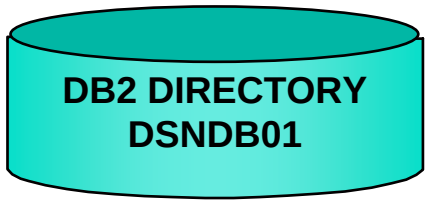
Log Ranges

SYSLGRNX

System Utilities

SYSUTILX

```
-DIS DATABASE(DSNDB01)
```




```
NAME     TYPE
-------- ----
DBD01    TS
SPT01    TS
SCT02    TS
SYSUTILX TS
SYSLGRNX TS
DSNSCT02 IX
DSNSPT01 IX
DSNSPT02 IX
DSNLUX01 IX
DSNLUX02 IX
DSNLLX01 IX
DSNLLX02 IX
```

- Internal DB2 system tables that describe:
  - DB2 Data Structure
  - How the data is stored
  - How DB2 can access the data
- Internal formats not accessible via DML
- Populated/Updated by DDL, bind processes, and utility operations
- VSAM datasets allocated by installation job DSNTIJIN

Directory Objects
Linkage to Operational Structures

Database Descriptors

DBD01

Skeleton Cursor Tables

SCT02

Skeleton Package Tables

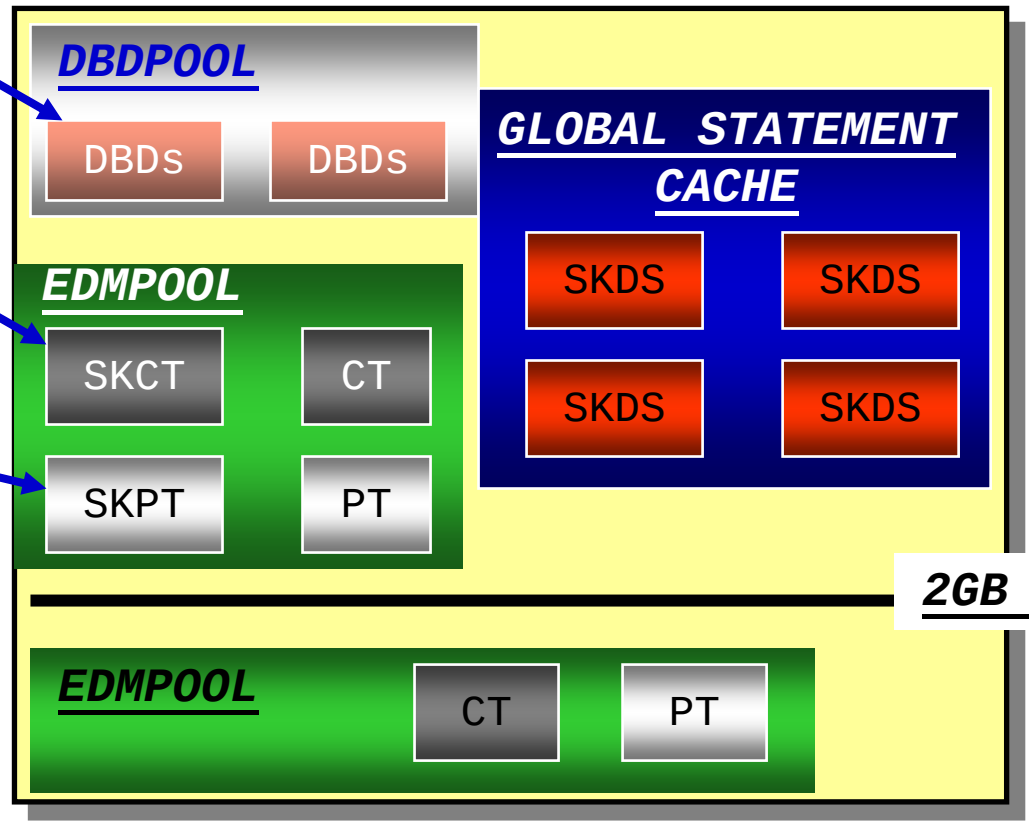
SPT01

Log Ranges

SYSLGRNX

System Utilities

SYSUTILX



Directory Contents
DBD01 – Database Descriptors

- What is the DBD?
  - Records the complete description of all objects defined within a single database
  - Complex hierarchical network of OBDs chained together
  - Each OBD is identified by a unique OBID (Object Identifier)
- Synchronized with DB2 catalog if everything is right
- No SQL access but information in the DB2 catalog (see table)
  - Access by DB2 internal processes
  - DISPLAY DATABASE shows lots of detail
- Create or updated as a result of DDL (CREATE, DROP, ALTER) or utility operations

Objects	Type	OBID Name	Catalog Table
Database	DBD	DBID	SYSDATABASE
Table Space	File Page set	OBID PSID	SYSTABLESPACE SYSTABLESPACE
Table	Record	OBID	SYSTABLES
Index space	Page set	ISOBID	SYSINDEXES
Index	Fan set	OBID	SYSINDEXES
RI Relationship	Fan set	RELOBID1 RELOBID2	SYSRELS SYSRELS
Auxiliary Relationship	OBID	AUXRELOBID	SYSAUXRELS

SYSDATABASE

```
Database Owner    Stogroup Buf Pool DBID
---1----v----2----v----3----v----4---v--
BMCQA1DB SLSTXM   BMCXXXSG BP0        287
```

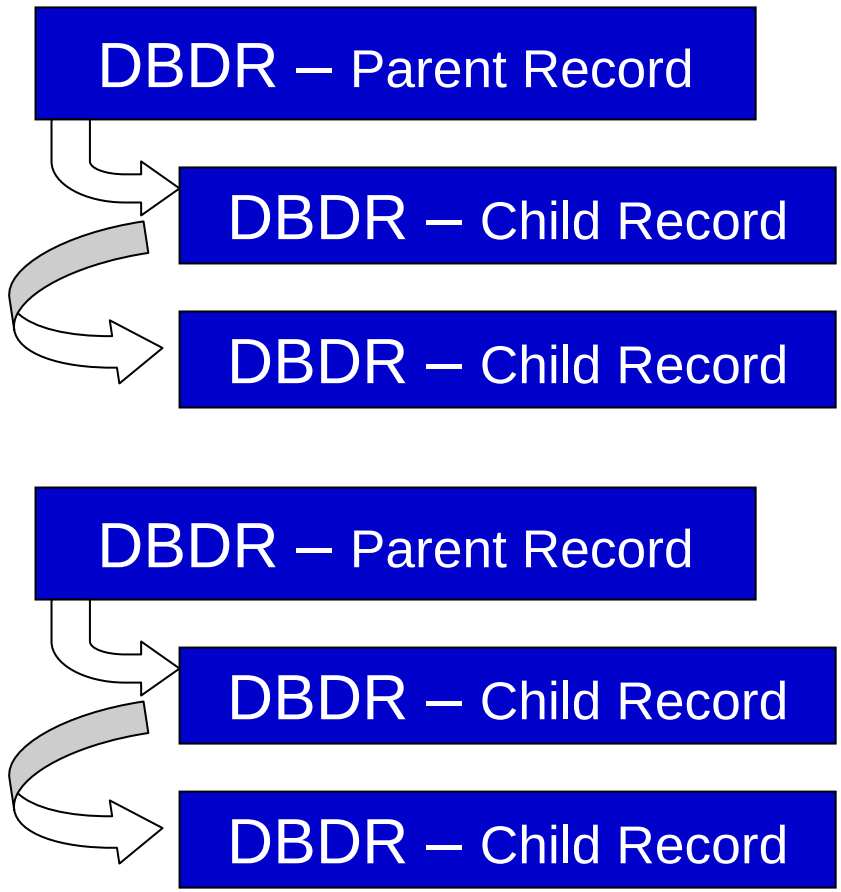
SYSTABLESPACE

```
TSNAME          OBID     PSID
--------+---------+------
WDATB1TS          216      217
WDATB9TS          219      220
```

bmcsoftware

Directory Contents
Database Descriptors On DASD

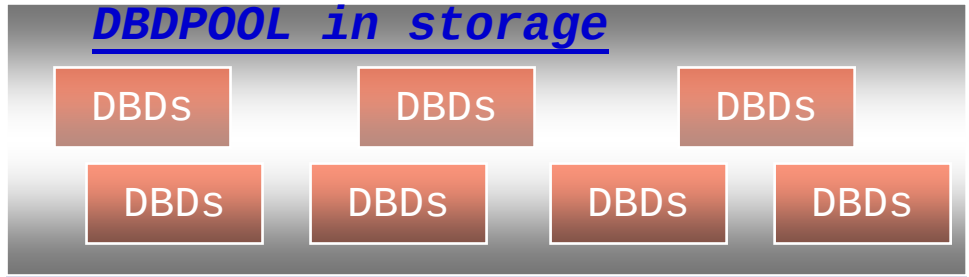
1. At CREATE DATABASE DBDR is created to hold this initial parent record. Access to the DBDR is via a hash. Most of the record is free space.
2. As objects are created entries (OBDs) are created in the record.
3. If tablespace or indexspace OBDs are dropped the space is immediately available for reuse
   - Table OBD space is not immediately reused
4. If the DBDR fills up DBDRs (Child Records) are created to hold the new OBDs.



DBD Pool
Operational Considerations

- Large DBDs can create operational problems
  - Large DBDs impact concurrency and logging
  - DDL against the DBD will lock out concurrent DDL activity and will also lock out dynamic SQL activity and some utility processing
  - Frequent changes to a large DBD will require more logging and increased I/O
- Space in the DBD for a dropped table is not immediately reusable
  - If you drop a table in a multi-table tablespace a reorganization is required to make this space available for subsequent DDL

- How many objects in the DBD?
  - Maximum concurrency use 1
    - Extra Administrative overhead
  - Multiple sources of design recommendations



DS Group : DSNDIA Member : DIA1
DB2 SSID : DIA1 9.1

Total EDM pool pages : 12867
DBD Pool pages : 25599 Pages Used : 386 (1.5 %)

Database	DBID	OBIDs for DBD	Pages Used by DBD	DBD Size (Bytes)	Free Space in DBD
DSNDB06	0006	895	60	242270	5.4 %
BMCPERF	0123	96	25	100940	3.8 %
BMCASU91	011A	248	23	92864	6.5 %
BMCUTIL	0106	137	13	52484	5.4 %
CRBMCDDB	018C	111	12	48446	8.0 %
BMCALP91	01A8	102	10	40370	12.6 %
AFLTESTA	018E	66	6	24218	12.6 %
BMCACT91	0119	63	5	20180	8.0 %
BMCAFD62	01A6	22	3	12104	27.3 %
BMCAFD61	0122	22	3	12104	27.3 %
DIA1	0100	8	1	4028	53.0 %

DBD01 Utility Considerations

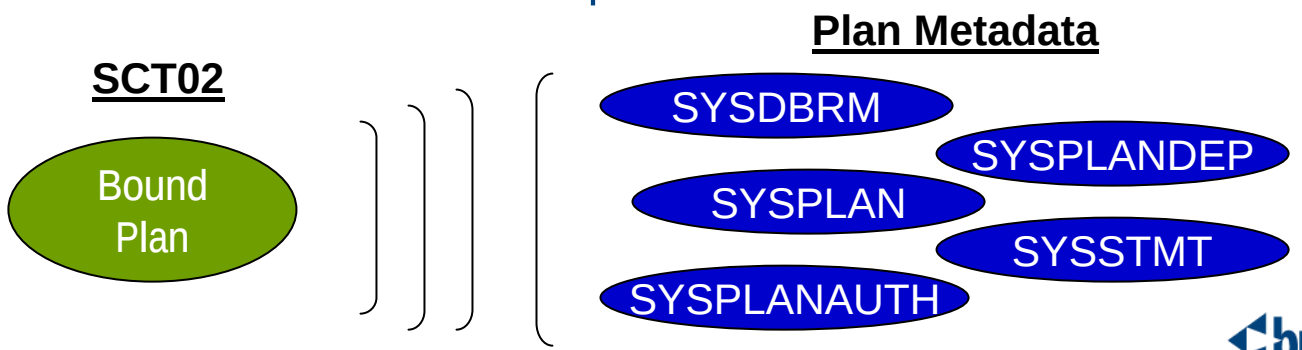
- COPY
  - As normal, DBD01 can and should be copied on a regular basis
  - DSNDB01.DBD01 copies are not recorded in SYSCOPY, instead they are recorded on the log
  - Incremental copy is not supported
- MODIFY RECOVERY
  - Eliminates entries in SYSCOPY, SYSLGRNX, and DBD01
- QUIESCE
  - Like the COPY, information about Quiesce is written to the log
- RECOVER
  - Catalog and Directory objects must be recovered in a specific sequence
    - Documented in multiple places
      - For directory objects
        1. DSNDB01.SYSUTILX and it's indexes
        3. DSNDB01.DBD01 (no indexes)
        6. DSNDB01.SYSLGRNX and it's indexes
    - Other directory objects come later

DBD01 Utility Considerations
More Thoughts

- REORG
  - Directory tablespaces should be reorganized when required
    - Usually not as frequently
    - Based on RUNSTATS metrics collected
- REPAIR DBD
  - Inconsistencies between DBDs and the catalog can occur
    - This stuff can be scary
  - Resolution process
    1. Run DSN1CHKR against DBD01
       - Scans for broken links, broken hash chains, and orphan records
    2. Start the space in UT (utility status)
    3. Run REPAIR DBD with TEST to identify any inconsistencies between directory and catalog
    4. If inconsistencies are found use the REPAIR DBD DIAGNOSE and the REPAIR DBD REBUILD options
       - Good time to be on the phone with the DB2 support team

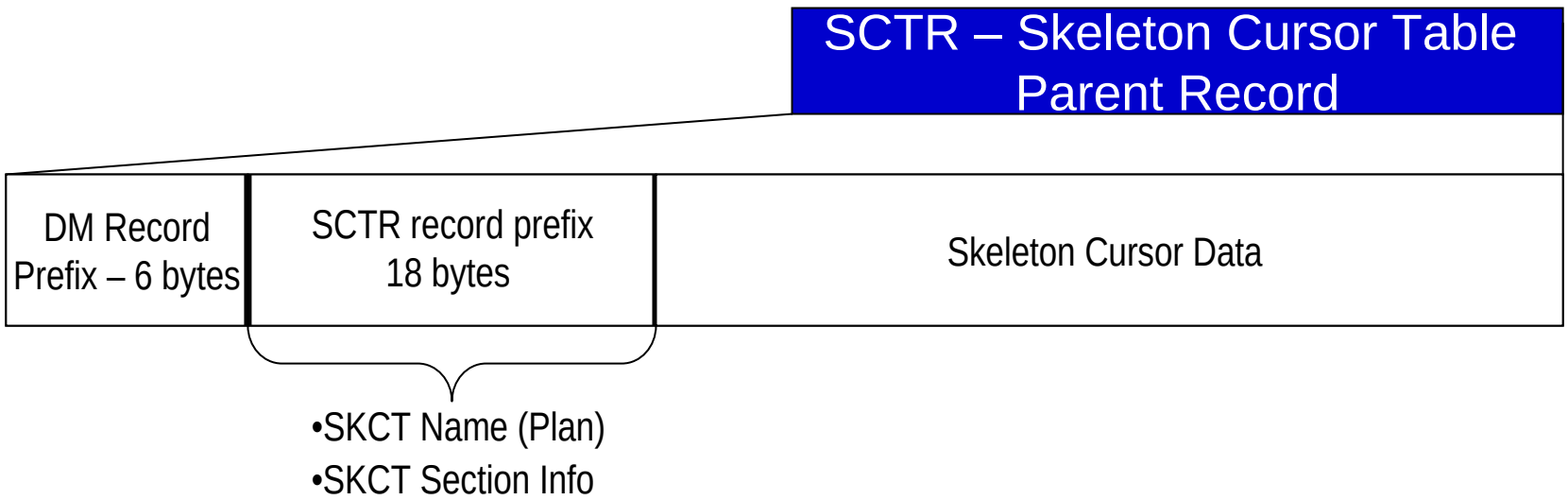
SCT02 – Skeleton Cursor Tables
An Introduction

- Structures containing operational access path information about plans
  - Packages is the recommended approach
  - Anybody still using DBRM-based plans?
- Skeleton Cursor Table (SKCT) -
  - Internal form of SQL statements contained in an application
  - Created and updated using BIND and REBIND PLAN command variations
  - Deleted using FREE PLAN command
- Loaded into EDM pool at plan execution
  - Concurrent plan user gets their own copy called the CT (Cursor Table)
  - DIRECTORY/CATALOG Relationship



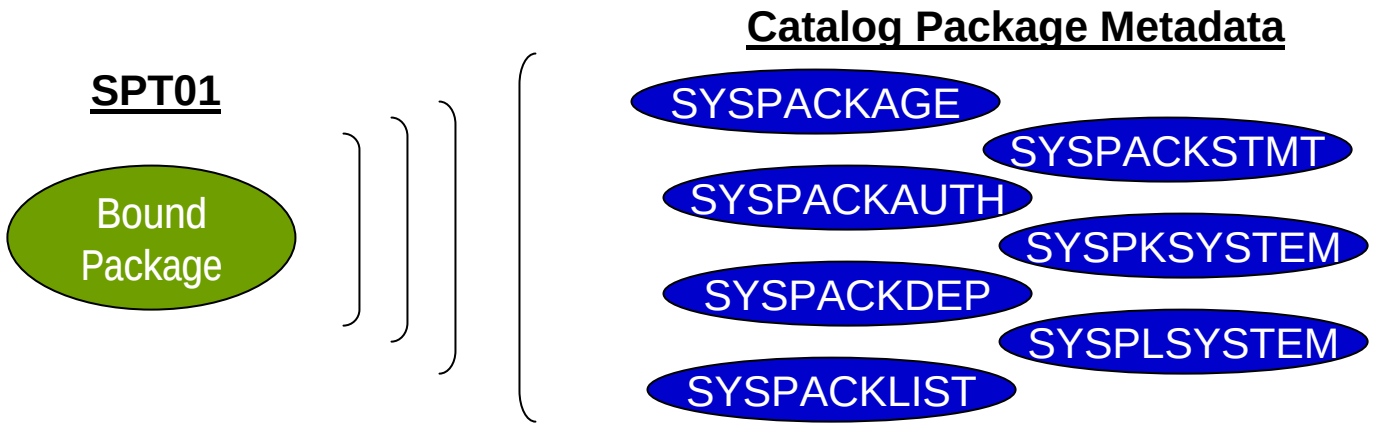
SCT02 Storage Implementation

- Pageset DSNDB01.SCT02
- Table SYSIBM.SCT02
- Accessed through one unique index
  - DSNSCT02 (SCTNAME,SCTSEC,SCTSEQ)
- Plans stored in one or more SCTR(s)
  - Header and multiple sections



SPT01 – Skeleton Package Tables
An Introduction

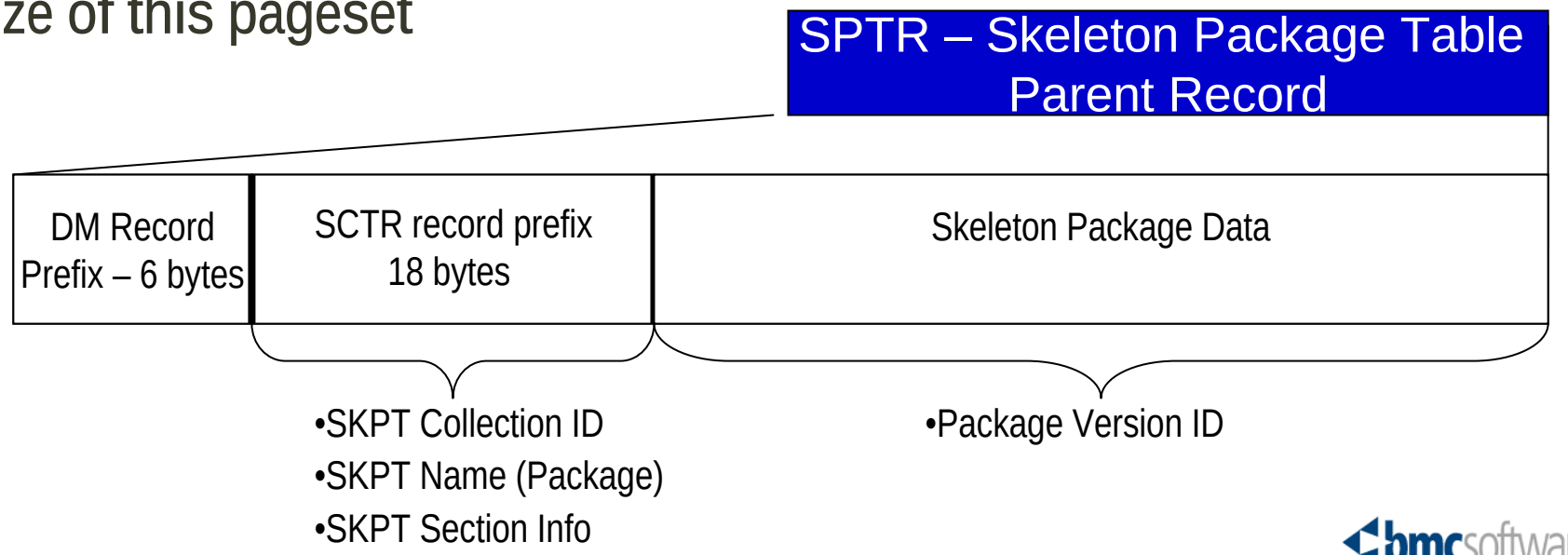
- Structures containing operational access path information about plans and packages
- SPT01 – Skeleton Package Table (SKPT)
  - Internal form of SQL statements contained in a package
  - Created and updated using BIND and REBIND PLAN command variations
  - Deleted using FREE PACKAGE command
- Plan user gets their own copy called the PT (Cursor Table)
- DIRECTORY/CATALOG Relationship



bmcsoftware

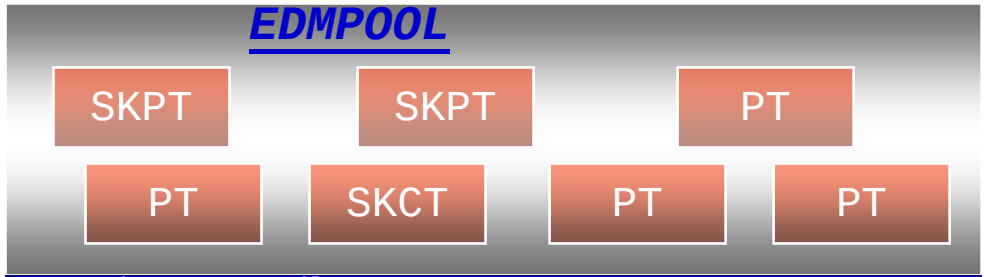
SPT01 Storage Implementation

- Pageset DSNDB01.SPT01
- Table SYSIBM.SPT01
- Accessed through two unique indexes
  - DSNSPT01 (SPTPID,SPTSEC,SPTSEQ)
  - DSNSPT02 (version, SPTID, SPTSEC, SPTSEQ)
- Package stored in one or more SPTRs
- DB2 9 Plan Stability feature can dramatically increase the size of this pageset



SPT01 Operational Considerations

- Packages should be your standard for all application development work
  - Reduces overall space required in the EDMPOOL
  - Bind processes are much simplified
- SPT01 will likely be your largest directory table
  - DB2 9 Plan Stability feature could triple the size of SPT01
  - Degree Any can increase the size of SPT01 by 50 to 70%
- Bind with Release (Commit) to free up packages from the EDMPOOL more quickly
- Proactively manage the contents of SPT01 to eliminate outdated packages
  - Package information is in catalog and can be used to identify packages that should be freed




```
   Expands  : S - Details
   DS group : DSNDIA          Member : DIA1
   DB2 SSID : DIA1 9.1
                      Total EDM pool pages   :    12867
                      Total used by SKPT, PT :       14  (   0.1 % )

                         SKPT       SKPT     Active    Average   Total
   Collection          Base Pages   Count    PT Count  PT  Size  PT Loads
   ------------------  ----------  --------  --------  --------  --------
+  DSNESPRR                     1         1         0      4028         2
    Skeleton Package Entries
    Pkg  : DSNESM68 Token : 149EEA901A79FE48 Users :         0
+  NULLID                       3         5         0     14156      2637
    Skeleton Package Entries
    Pkg  : DSN5OADM Token : 6941754D52424C73 Users :         0
    Pkg  : DSN5OPKG Token : 5142364A4C434C72 Users :         0
    Pkg  : SQLA1F00 Token : 4141414141614853 Users :         0
    Pkg  : SYSLH200 Token : 5359534C564C3031 Users :         0
    Pkg  : SYSSTAT  Token : 5359534C564C3031 Users :         0
+  RXD2                         1         1         0      4276         2
```

SCT02 & SPT01 Utility Considerations

- COPY
  - Create backups of SCT02 and SPT01 on a regular basis
- REORG
  - Use SHRLEVEL REFERENCE for directory objects
  - Make image copies before and after the reorganization
  - When should you REORG?
    - For these two spaces use the same metrics you might for applications tablespaces and indexspaces
      - PERCDROP in SYSIBM.SYSTABLESPART
      - LEAFNEAR/LEAFFAR in SYSIBM.SYSINDEXPART
      - NEARINDREF/FARINDREF in SYSIBM.SYSTABLEPART
      - CLUSTERRATIO (not an exhaustive list)
    - Reorg these tablespaces when associated catalog tables are reorganized
      - DSNDB01.SPT01 when DSNDB06.SYSPKAGE is reorganized
      - DSNDB01.SCT02 when DSNDB06.SYSPLAN is reorganized

DBD01, SCT02, SPT01 Performance Considerations

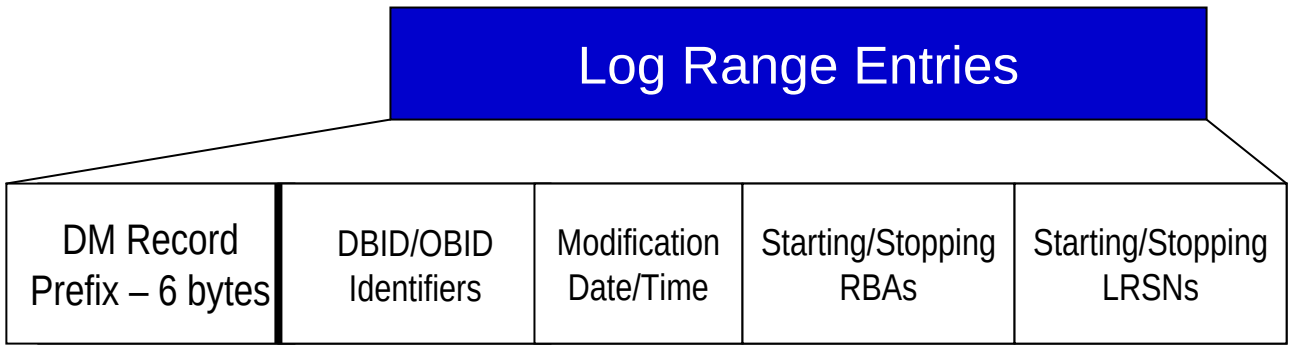
- Physical size of the pagesets for these structures are all calculated during DB2 installation using values you plug into a formula
  - Beware the issue in DB2 9 with the new PLANMGMT options that can dramatically increase the size of SPT01
- Operational structures like the DBD Pool and EDM Pool are controlled by DSNZPARM values and these are also calculated during initial install
- Monitor the statistics related to how often these structures have to be loaded from disk into the different pools
  - Look at hit ratios and if too low consider increasing the size of the pools

SYSLGRNX – System Log Ranges
An Introduction

- Stores recovery log ranges that record the time an index space defined with COPY YES or a table space was open for updates
- SYSLGRNX – System Log Range Table
  - Records intervals on the DB2 log where updates for an object may occur
    - When a DB2 object is opened for update and subsequently closed that interval is recorded
  - Primary goal is to minimize amount of log data that must be processed for a recovery
  - Part of recovery assets required to complete the recovery process
    - DB2 active and archive log datasets that map to SYSLGRNX log ranges
    - Image copy datasets required

SYSLGRNX Storage Implementation

- Pageset DSNDB01.SYSLGRNX
- Table name SYSIBM.SYSLGRNX
- Accessed through two indexes
  - DSNLLX01 (LGRDBID,LGRPSID,LGRPART,LGRMEMB, LGRSLRSN)
  - DSNLLX02 (LGRDBID,LGRPSID,LGRSLRSN)
- Entries stored in a fixed length record



SYSLGRNX – What do they Look Like?

- Use the RECOVERY option of the REPORT utility to see the log range entries for a specific object
- An example:

```
DSNU586I  ) 277 11:56:29.14 DSNUPSUM - REPORT RECOVERY TABLESPACE DBUV0101.TPUV0101 SUMMARY
DSNU588I  ) 277 11:56:29.14 DSNUPSUM - NO DATA TO BE REPORTED

/DSNU583I  ) 277 11:56:29.14 DSNUPPLR - SYSLGRNX ROWS FROM REPORT RECOVERY
             FOR TABLESPACE DBUV0101.TPUV0101
```

UCDATE	UCTIME	START RBA	STOP RBA	START LRSN	STOP LRSN	PARTITION	MEMBER ID
100406	11541904	0000374F86EC	00003752195A	BF8110CF0ADF	BF8110D0EB95	0001	0000
100406	11541916	0000374FB0E1	00003752195A	BF8110CF26BE	BF8110D0EC6F	0002	0000
100406	11541929	0000374FDACA	00003752195A	BF8110CF4606	BF8110D0ECEE	0003	0000
100406	11541940	000037500483	00003752195A	BF8110CF6209	BF8110D0ED64	0004	0000
100406	11541952	000037502E23	00003752195A	BF8110CF7F04	BF8110D0EE47	0005	0000
100406	11541964	00003750582E	00003752195A	BF8110CF9AFD	BF8110D0EED8	0006	0000
100406	11541975	0000375081E7	00003752195A	BF8110CFB7D6	BF8110D0EF51	0007	0000
100406	11541987	00003750AB87	00003752195A	BF8110CFD3C4	BF8110D0EFCC	0008	0000
100406	11541998	00003750D540	00003752195A	BF8110CFEFD4	BF8110D0F052	0009	0000
100406	11542010	00003750FEE0	00003752195A	BF8110D00D2F	BF8110D0F0D6	0010	0000

SYSLGRNX Operational Considerations

- Entries will grow over time and the table will become quite large
- Use the MODIFY RECOVERY to manage SYSLGRNX entries
  - Deletes SYSLGRNX and SYSCOPY rows from a single partition or an entire tablespace (DSNUM option)
  - SYSLGRNX rows where there are no SYSCOPY rows
  - Recovery rows for indexes
  - Also reclaims space in the DBD
- Sample Utility Control Statement
  - Deletes SYSCOPY and SYSLGRNX records written before Sept 10, 2002

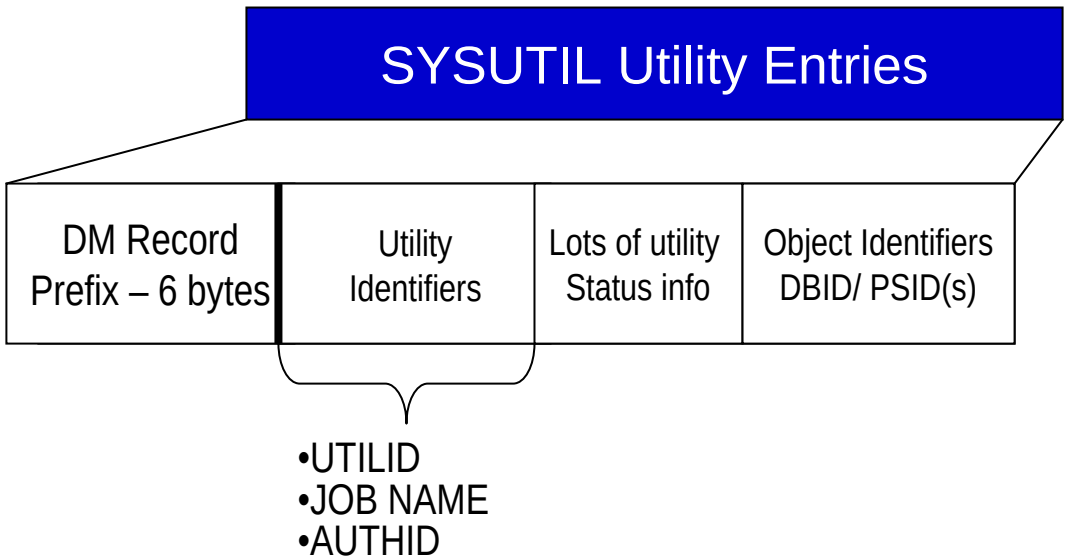
```
MODIFY RECOVERY TABLESPACE DSN8D91A.DSN8S91D DELETE DATE(20020910)
```

SYSUTILX – DB2 Utilities Register
An Introduction

- Stores status information about DB2 utilities that are active or stopped
- Here's what happens
  1. Utility begins and is registered in the SYSUTIL table
  2. As the utility moves from phase to phase a new record is added for each new utility execution step.
  3. When the utility completes all the corresponding entries in the SYSUTIL table are deleted.
  4. If the utility stops during execution all the records remain till the problem is resolved

SYSLGRNX Storage Implementation

- Pageset DSNDB01.SYSUTILX
- Table name SYSIBM.SYSUTIL and SYSIBM.SYSUTILX
- Accessed through two unique indexes
  - DSNLUX01 (USUUID) – UTILID index built on SYSUTIL)
  - DSNLUX02 (UTILID,SEQNO) – built on SYSUTILX
- Entries stored in a fixed length record
  - SYSUTILX is overflow for SYSUTIL records



DISPLAY UTILITY Command
What Will You See?

- DISPLAY UTILITY command is the easiest way to look at what utilities are running in your system.
  - DISPLAY UTILITY (UTILID) or * or partial key
- Output will vary based on the type of utility process is running
- An example:

```
DSNU100I -DB1G DSNUGDIS USER = SAMPID
                        MEMBER = DB2G
                        UTILID = CHKIX1
                        PROCESSING UTILITY STATEMENT 8
                        UTILITY = CHECK
                        PHASE = UNLOAD  COUNT = 0
                        STATUS = STOPPED
DSN9022I -DB1G DSNUGCC  '-DB1G DISPLAY UTILITY' NORMAL COMPLETION
```

SYSUTILX Operational Considerations

- Unique characteristics for this directory tablespace
  - One of three catalog/directory objects where copies are not recorded in SYSCOPY but on the log
- Copy considerations
  - To copy DSNDB01.SYSUTILX the copy must be the only utility in the job step
  - A SHRLEVEL REFERENCE Copy requires no other utilities running in that data sharing group
  - An “exclusive” utility that can interrupt other tasks that may be running
- No reorganizations possible
- Must be the first tablespace recovered if recovering multiple catalog/directory objects
  - Recovery is not restartable so SYSUTILX must be re-initialized
  - May require manual resolution of objects that are in “restricted” status because of a “stopped” utility

Questions?