DB2 Directory – What, How, When, and Where

Bill Arledge, Consulting Product Manager
BMC Software Inc.
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 DSNnnnnnn**
  - Implicitly defined databases supported in DB2 9
DB2 System Databases
“Catalog” Tablespaces & Tables

- 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
<table>
<thead>
<tr>
<th>Directory Structures</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBD01</strong> Database Descriptors</td>
</tr>
<tr>
<td>- A run-time operational structure that identifies all DB2 objects created in a specific database</td>
</tr>
<tr>
<td>- Internal structure not visible to the “naked” eye</td>
</tr>
<tr>
<td><strong>SCT02</strong> Skeleton Cursor Table</td>
</tr>
<tr>
<td>- Operational run-time structure with access path information about a DBRM-based plan</td>
</tr>
<tr>
<td><strong>SPT01</strong> Skeleton Package Table</td>
</tr>
<tr>
<td>- Operational run-time structure with access path information about packages</td>
</tr>
<tr>
<td><strong>SYSLGRNX</strong> SYSLGRNX – Recovery Log Ranges</td>
</tr>
<tr>
<td>- Provides details by DB2 object about log ranges required for recovery processing</td>
</tr>
<tr>
<td><strong>SYSUTILX</strong> DB2 In-flight Utilities Register</td>
</tr>
<tr>
<td>- Contains entries about DB2 utilities currently running or abnormally terminated/stopped</td>
</tr>
</tbody>
</table>
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)**

DB2 DIRECTORY
DSNDB01

- 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

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBD01</td>
<td>TS</td>
</tr>
<tr>
<td>SPT01</td>
<td>TS</td>
</tr>
<tr>
<td>SCT02</td>
<td>TS</td>
</tr>
<tr>
<td>SYSUTILX</td>
<td>TS</td>
</tr>
<tr>
<td>SYSLGRNX</td>
<td>TS</td>
</tr>
<tr>
<td>DSNCT02</td>
<td>IX</td>
</tr>
<tr>
<td>DSNPT01</td>
<td>IX</td>
</tr>
<tr>
<td>DSNPT02</td>
<td>IX</td>
</tr>
<tr>
<td>DSNLUX01</td>
<td>IX</td>
</tr>
<tr>
<td>DSNLUX02</td>
<td>IX</td>
</tr>
<tr>
<td>DSNLLX01</td>
<td>IX</td>
</tr>
<tr>
<td>DSNLLX02</td>
<td>IX</td>
</tr>
</tbody>
</table>
Directory Objects
Linkage to Operational Structures

- **Database Descriptors**
  - DBD01

- **Skeleton Cursor Tables**
  - SCT02

- **Skeleton Package Tables**
  - SPT01

- **Log Ranges**
  - SYSLGRNX

- **System Utilities**
  - SYSUTILX

---

**DBM1 - DB2 Database Services**

- **DBDPOOL**
  - DBDs

- **EDMPOOL**
  - SKCT
  - SKPT
  - CT
  - PT

- **GLOBAL STATEMENT**
  - CACHE
    - SKDS

- **2GB Bar**
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

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>OBID Name</th>
<th>Catalog Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>DBD</td>
<td>DBID</td>
<td>SYSDATABASE</td>
</tr>
<tr>
<td>Table Space</td>
<td>File</td>
<td>OBID</td>
<td>SYSTABLESPACE</td>
</tr>
<tr>
<td></td>
<td>Page set</td>
<td>PSID</td>
<td>SYSTABLESPACE</td>
</tr>
<tr>
<td>Table</td>
<td>Record</td>
<td>OBID</td>
<td>SYSTABLES</td>
</tr>
<tr>
<td>Index space</td>
<td>Page set</td>
<td>ISOBID</td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td>Index</td>
<td>Fan set</td>
<td>OBID</td>
<td>SYSINDEXES</td>
</tr>
<tr>
<td>RI Relationship</td>
<td>Fan set</td>
<td>RELOBID1</td>
<td>SYSRELS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RELOBID2</td>
<td>SYSRELS</td>
</tr>
<tr>
<td>Auxiliary</td>
<td>OBID</td>
<td>AUXRELOBID</td>
<td>SYSAUXRELS</td>
</tr>
</tbody>
</table>

SYSDATABASE

<table>
<thead>
<tr>
<th>Database Owner</th>
<th>Stogroup</th>
<th>Buf Pool</th>
<th>DBID</th>
</tr>
</thead>
<tbody>
<tr>
<td>---1---v---2---v---3---v---4---v---</td>
<td>BMCQA1DB</td>
<td>SLSTXM</td>
<td>287</td>
</tr>
</tbody>
</table>

SYSTABLESPACE

<table>
<thead>
<tr>
<th>TSTYPE</th>
<th>OBID</th>
<th>PSID</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDATB1TS</td>
<td>216</td>
<td>217</td>
</tr>
<tr>
<td>WDATB9TS</td>
<td>219</td>
<td>220</td>
</tr>
</tbody>
</table>
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.
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 tablesapce 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
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
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

Plan Metadata

- SCT02
- Bound Plan
- SYSDBRM
- SYSPLAN
- SYSPLANAUTH
- SYSPLANDEP
- SYSSTMT

bmcsoftware
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

SCTR – Skeleton Cursor Table

Parent Record

DM Record Prefix – 6 bytes

SCTR record prefix 18 bytes

Skeleton Cursor Data

- SKCT Name (Plan)
- SKCT Section Info
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

![Diagram showing the relationship between SPT01 and Catalog Package Metadata](image-url)

- Bound Package
- SYSPACKAGE
- SYSPACKAUTH
- SYSPACKSTMT
- SYSPACKDEP
- SYSPACKAUTH
- SYSSYSTEM
- SYSPLSYSTEM
- SYSPACKLIST
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

**SPTR – Skeleton Package Table**

**Parent Record**

- **DM Record**
  - Prefix – 6 bytes

- **SCTR record prefix**
  - 18 bytes

- **Skeleton Package Data**
  - SKPT Collection ID
  - SKPT Name (Package)
  - SKPT Section Info
  - Package Version ID
### 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

---

**EDMPOOL**

<table>
<thead>
<tr>
<th>Collection</th>
<th>SKPT</th>
<th>Base Pages</th>
<th>SKPT Count</th>
<th>Active PT Count</th>
<th>Average Size</th>
<th>Total PT Loads</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSNESPPRRA</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4028</td>
<td>2</td>
</tr>
<tr>
<td>Skeleton Package Entries</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pkg : DSNESM68 Token : 149EEA901A79FE48 Users :</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NULLID</td>
<td>3</td>
<td>5</td>
<td>0</td>
<td>14156</td>
<td>2637</td>
<td></td>
</tr>
<tr>
<td>Skeleton Package Entries</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pkg : DSNSOADM Token : 6941754D52424C73 Users :</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pkg : DSNSOPKG Token : 5142364A4C343C72 Users :</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pkg : SQLA1F00 Token : 41414141414144853 Users :</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pkg : SYSLH200 Token : 5359534C64C3031 Users :</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pkg : SYSSTAT Token : 5359534C64C3031 Users :</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RXD2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4276</td>
<td>2</td>
</tr>
</tbody>
</table>

---

**Expands : S | Details**

- **Group : DSNNDIA**
- **Member : DIA1**
- **DB2 SSID : DIA1 9.1**
- **Total EDM pool pages : 12867**
- **Total used by SKPT, PT : 14 ( 0.1 %)**

---

**bmc software**
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.SYSTABLESPART
      – 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
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

Log Range Entries

<table>
<thead>
<tr>
<th>DM Record Prefix – 6 bytes</th>
<th>DBID/OBID Identifiers</th>
<th>Modification Date/Time</th>
<th>Starting/Stopping RBAs</th>
<th>Starting/Stopping LRSNs</th>
</tr>
</thead>
</table>

bmcsoftware
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:

<table>
<thead>
<tr>
<th>UPDATE</th>
<th>UCTIME</th>
<th>START RBA</th>
<th>STOP RBA</th>
<th>START LRSN</th>
<th>STOP LRSN</th>
<th>PARTITION</th>
<th>MEMBER ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>100406</td>
<td>11541904</td>
<td>0000374F86EC</td>
<td>00003752195A</td>
<td>BF8110CF0ADF</td>
<td>BF8110D0EB95</td>
<td>0001</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541916</td>
<td>0000374FB0E1</td>
<td>00003752195A</td>
<td>BF8110CF86BE</td>
<td>BF8110D0EC6F</td>
<td>0002</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541929</td>
<td>0000374FDACA</td>
<td>00003752195A</td>
<td>BF8110CF4606</td>
<td>BF8110D0ECEE</td>
<td>0003</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541940</td>
<td>000037500483</td>
<td>00003752195A</td>
<td>BF8110CF6209</td>
<td>BF8110D0ED64</td>
<td>0004</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541952</td>
<td>000037502E23</td>
<td>00003752195A</td>
<td>BF8110CF7F04</td>
<td>BF8110D0EE47</td>
<td>0005</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541964</td>
<td>00003750582E</td>
<td>00003752195A</td>
<td>BF8110CF9AFD</td>
<td>BF8110D0EE8</td>
<td>0006</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541975</td>
<td>0000375081E7</td>
<td>00003752195A</td>
<td>BF8110CF87D6</td>
<td>BF8110D0EF51</td>
<td>0007</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541987</td>
<td>00003750AB87</td>
<td>00003752195A</td>
<td>BF8110CFD3C4</td>
<td>BF8110D0EFCC</td>
<td>0008</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11541998</td>
<td>00003750D546</td>
<td>00003752195A</td>
<td>BF8110CF3F0D4</td>
<td>BF8110D0F052</td>
<td>0009</td>
<td>0000</td>
</tr>
<tr>
<td>100406</td>
<td>11542010</td>
<td>00003750FEE0</td>
<td>00003752195A</td>
<td>BF8110D0D02F</td>
<td>BF8110D0F0D6</td>
<td>0010</td>
<td>0000</td>
</tr>
</tbody>
</table>
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 (USUID) – UTILID index built on SYSUTIL
  - DSNLUX02 (UTILID,SEQNO) – built on SYSUTILX
- Entries stored in a fixed length record
  - SYSUTILX is overflow for SYSUTIL records

SYSUTIL Utility Entries

<table>
<thead>
<tr>
<th>DM Record Prefix – 6 bytes</th>
<th>Utility Identifiers</th>
<th>Lots of utility Status info</th>
<th>Object Identifiers DBID/ PSID(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• UTILID</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• JOB NAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• AUTHID</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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?