Back to Basics: All you ever wanted to know about the Db2 storage hierarchy

Payton Lawson
Software Developer
payton.lawson@ibm.com

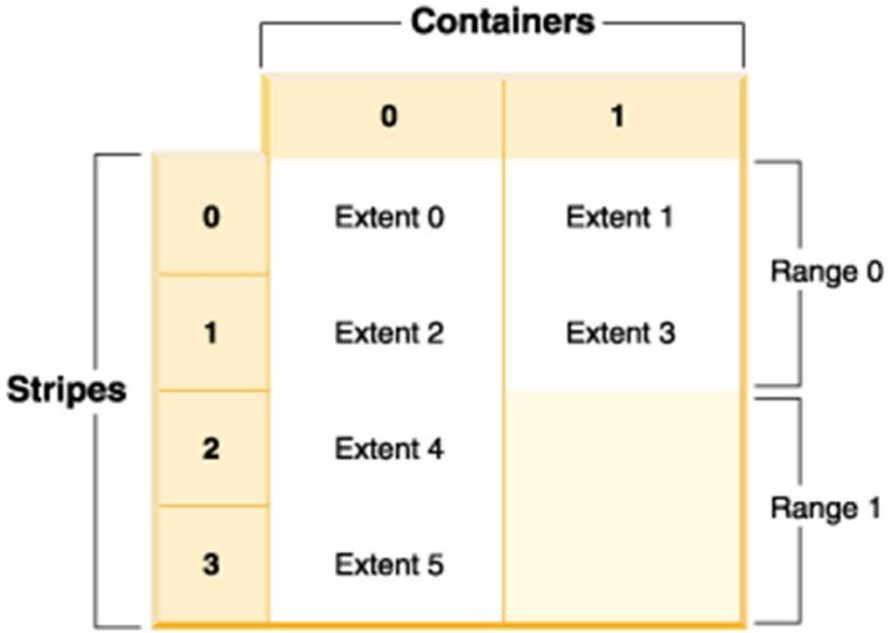
IBM

Table Space Fundamentals

The Basics: Table Spaces

Table spaces organize database objects, such as tables and indexes, into logical storage groups. Each table space is made up of one or more containers.

Containers are the physical location where your information is stored. A container can be a directory, a device, or a file.

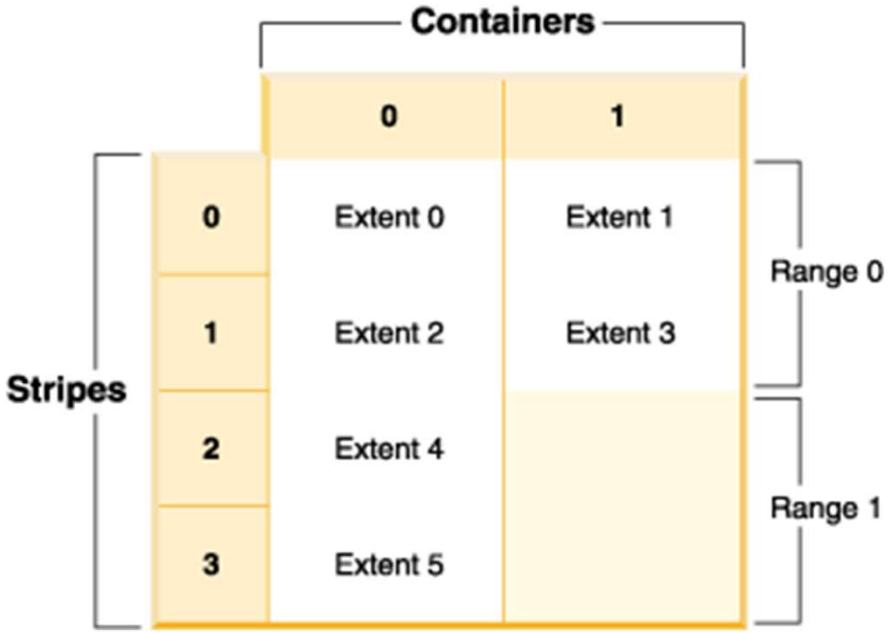


The Basics: Table Spaces

An extent is a contiguous block of pages. The number of pages representing an extent is configurable and can range from 2 – 256.

A stripe is a contiguous block of extents spanning distinct containers. Not to be confused with disk stripes.

A range is a contiguous range of stripes, all of which contain the same set of containers.



The Basics: SMS vs DMS Table Spaces

SMS – System Managed Storage

```
CREATE TABLESPACE … MANAGED BY SYSTEM
USING <container-clause> …
```

- Containers are filesystem directories
- Each directory contains multiple files representing objects (tables, indexes, etc.)
- Storage allocation happens on demand. Controlled by file system manager

Both SMS and DMS are deprecated for user-defined permanent table spaces.

DMS – Database Managed Storage

```
CREATE TABLESPACE … MANAGED BY DATABASE
USING <container-clause> …
```

- Containers are files or raw devices
- Data resides in space pre-allocated by the database
- Can be defined as REGULAR or LARGE, which affects the maximum size of the table space
  - Example: for a 32K page size:
    - The maximum size of a DMS REGULAR table space is **512 GB**
    - The maximum size of a DMS LARGE table space is **64 TB**

The Basics

Temporary Table Spaces

Temporary table spaces contain temporary tables

System Temporary Table Spaces

- Hold temporary data required by the database manager while performing operations such as sorts or joins
- Must be at least one system temporary table space
- By default, one system temporary table space called TEMPSPACE1 is created during database creation

User Temporary Table Spaces

- Hold temporary data from tables created with a DECLARE / CREATE GLOBAL TEMPORARY TABLE statement
- Not created by default during database creation

- When new temporary objects are needed, the query optimizer will choose an appropriate page size for this object.
- If a temporary table space of the required page size does not exist, the query will fail.
- If there is more than one temporary table space with that page size, then the table space will be chosen in a round-robin fashion
- Not recommended to have more than one temporary table space with the same page size

The Basics

Special Table Spaces

SYSCATSPACE

- A catalog table space that contains all of the system catalog tables for the database
- Automatically created
- Cannot be dropped

The Basics

Special Table Spaces

SYSTOOLSPACE

User data table space used by Db2 admin tools and some SQL admin routines for storing historical data and configuration information.

Created automatically when needed in most cases.

The SYSTOOLSPACE table space must be manually created before using any of the following:

- db2look command
- administrative task scheduler
- ALTOBJ
- ADMIN_COPY_SCHEMA, and
- ADMIN_DROP_SCHEMA

```
CREATE TABLESPACE SYSTOOLSPACE IN IBMCATGROUP
   MANAGED BY AUTOMATIC STORAGE USING
   STOGROUP IBMSTOGROUP EXTENTSIZE 4
```

The Basics

Special Table Spaces

SYSTOOLSTMPSPACE

User temporary table space used by:
- REORGCHK_TB_STATS
- REORGCHK_IX_STATS, and
- ADMIN_CMD procedures

for storing temporary data

Created automatically when needed in most cases.

The SYSTOOLSTMPSPACE table space must be manually created before using ADMIN_CMD.

```
CREATE USER TEMPORARY TABLESPACE SYSTOOLSTMPSPACE
   IN IBMCATGROUP MANAGED BY AUTOMATIC STORAGE
   USING STOGROUP IBMSTOGROUP EXTENTSIZE 4
```

The Basics

Special Table Spaces

Notes on SYSTOOLSPACE and SYSTOOLSTMPSPACE:

- If DB2_WORKLOAD is set to SAP, neither table space will be created automatically if the database is not automatic storage enabled
- Both table spaces should be created automatically for new active databases if they have been active for two hours
  - AUTO_MAINT / AUTO_TBL_MAINT / AUTO_RUNSTATS are ON by default on all new databases
  - Evaluated every two hours
  - Internals will drive creation of these table spaces

The Basics: Separate Table Spaces for Indexes and LOBs

Specify a separate table space for index data using the “INDEX IN” clause of the “CREATE TABLE” statement:

```
CREATE TABLE … INDEX IN <tablespace-name>
```

- Table space must be SMS if the table has data in SMS table spaces; must be DMS if the table has data in DMS table spaces.
- For a nonpartitioned index on a partitioned table, this can also be specified as the IN clause of CREATE INDEX

Specify a separate table space for LONG VARCHAR/LOB data using the “LONG IN” clause of the “CREATE TABLE” statement:

```
CREATE TABLE … LONG IN <tablespace-name>
```

- Option is only allowed if it specifies a DMS or automatic storage table space
- For partitioned tables, can provide a list of table spaces to be used in a round-robin order

Automatic Storage and Storage Groups

Automatic Storage and Storage Groups

Automatic Storage

Automatic storage greatly simplifies storage management for table spaces.

```
CREATE TABLESPACE … MANAGED BY AUTOMATIC STORAGE
USING <container-clause> …
```

Common misconception: automatic storage is not another table space type. It directs the database manager to select the appropriate storage model and properties based on the type of tablespace being created.

Db2 will take care of the placement, naming, allocation, and future growth of table space containers automatically

- Automatic SMS is only supported with temporary table spaces
- Automatic (DMS for data, SMS for temps) is the only table space type supported in pureScale

Automatic Storage and Storage Groups

Storage Groups

A named set of storage paths where data can be stored

When you create a database enabled for AUTOMATIC STORAGE (without the “AUTOMATIC STORAGE NO” clause), a default storage group named IBMSTOGROUP is automatically created

Storage groups can only be used by automatic storage table spaces.

A table space can be associated with only one storage group, but a storage group can have multiple table space associations

Storage groups can be defined using the following SQL:

```
CREATE STOGROUP <stogroup-name> ON <storage-paths>
```

- `SET AS DEFAULT` - Designate that any new table space managed by automatic storage with no explicit `USING STOGROUP` clause be associated with this storage group

Ensure that the storage paths added to a storage group share similar media characteristics. If the media characteristics are dissimilar, specify a value that represents an average for OVERHEAD and DEVICE READ RATE

Automatic Storage and Storage Groups

Modifying Storage Groups

`ALTER STOGROUP <stogroup-name> …`

`ADD <storage-paths>`

- Specifies one or more new storage paths are to be added to the storage group
- For multipartition environments, the storage path must exist on each database partition
- This new storage path will be used only when the containers for the current stripe run out of space. To use them immediately, use `ALTER TABLESPACE … REBALANCE`

`DROP <storage-paths>`

- Specifies one or more storage paths are to be removed from the storage group
- If table spaces are actively using a storage path being dropped, then the state of the storage path is changed from "In Use" to "Drop Pending" and future use of the storage path will be prevented
- Need to issue `ALTER TABLESPACE … REBALANCE` to move allocated extents off the dropped containers
- **Not supported in pureScale**

Migrating to Automatic Storage

Method 1: Table Space Alter

- `ALTER TABLESPACE … MANAGED BY AUTOMATIC STORAGE USING STOGROUP …` followed by `ALTER TABLESPACE … REBALANCE`
- Works with DMS table spaces only
- Fully online

Method 2: Redirected Restore

- `SET TABLESPACE CONTAINERS … USING AUTOMATIC STORAGE` followed by `RESTORE then ALTER TABLESPACE … MANAGED BY AUTOMATIC STORAGE`
- Works with DMS table spaces only
- Can also be used to convert from raw to filesystem-based containers

Method 3: Schema Transport

- `RESTORE … TRANSPORT INTO`
- Useful when converting SMS catalog table space to automatic DMS
- http://www-01.ibm.com/support/docview.wss?uid=swg21984655

Method 4: Data Unload

- Unload data, create a new automatic table space, reload with data
- Works with any table space type
- db2move, ADMIN_MOVE_TABLE, db2look/EXPORT/LOAD
- Requires manual work

Reclaimable Storage

What is Reclaimable Storage?

- A change to the on-disk format of non-temporary DMS/AS table spaces that allows improved space management

- Table spaces created before v9.7 must be recreated in this new format. In-place or automatic migration of existing tablespaces is not feasible and has not been implemented.
  - Unload data, create a new automatic table space, reload with data
  - db2move, ADMIN_MOVE_TABLE, db2look/EXPORT/LOAD

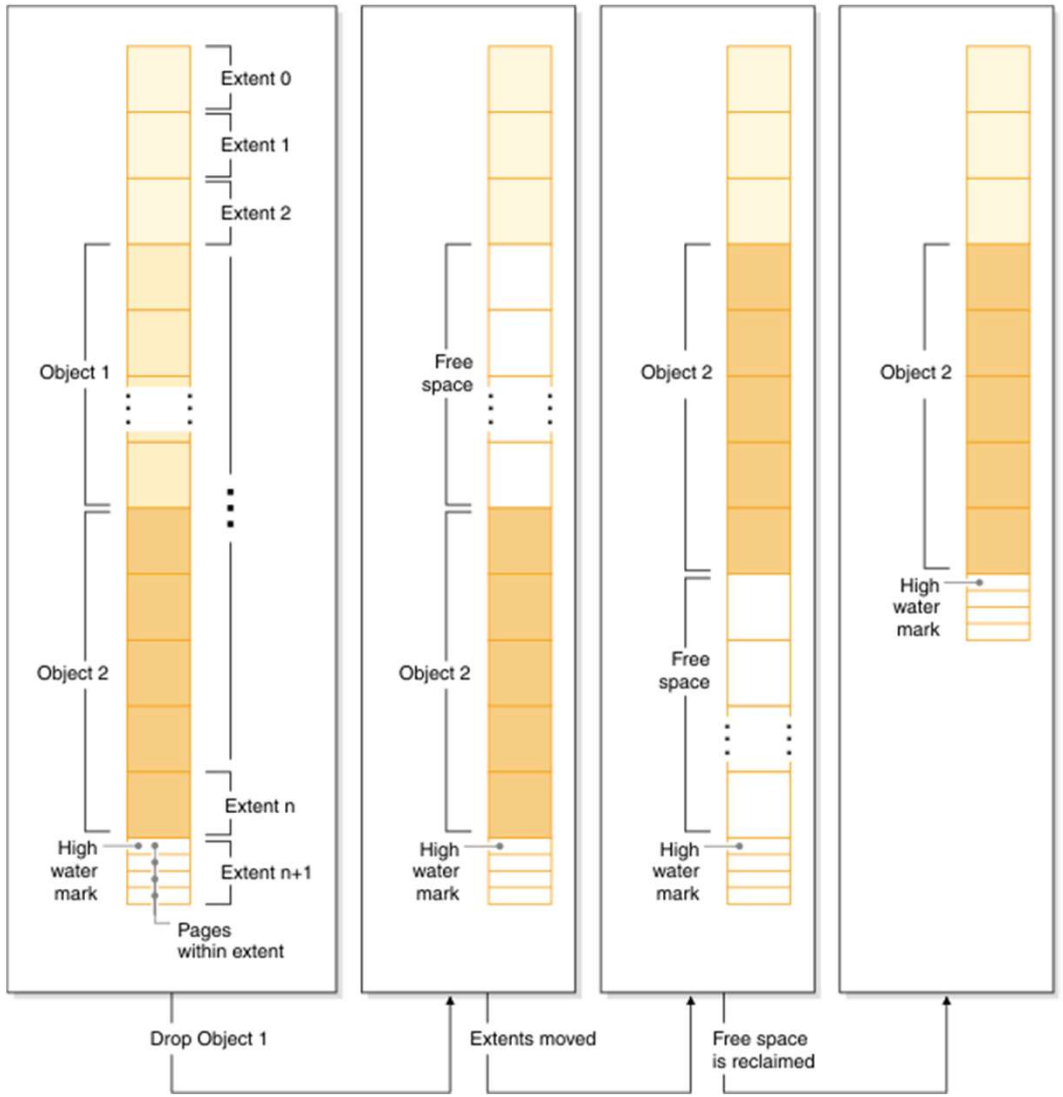
Extent Movement

- Allows the relocation of extents on disk to fill holes. Analogous to file system defrag
- One extent reclaim per table space, one mover thread EDU per extent reclaim
- Page metadata changes – the page is physically copied to a different location in the table space
- The table space map remains unchanged
- Performance not critical, usually negligible impact
- Incompatible with operations such as backup, load, and others; extent reclaim will yield to other operations and pause automatically

Extent Movement

When an object is dropped below the high water mark, we are left with free extents in the middle of the table space.

By moving the extents for Object 2 lower in the table space, we can reduce the size of the containers and release unused space back to the file system.



Reclaimable Storage

Automatic Storage supports:

- Container reduction only
- Lower high water mark only
- Lower high water mark and reduce containers by a specific amount
- Lower high water mark and reduce containers the maximum amount possible

The two additional modes are supported under AS because storage is managed and configured uniformly.

DMS supports:

- Container reduction only
- Lower high water mark only

Reclaimable Storage in pureScale

Starting in Version 11.5, reclaimable storage operations are allowed in Db2 pureScale environments by default (the DB2_ENABLE_PS_EXTENT_RECLAIM registry variable defaults to ON)

- One extent reclaim per table space, one mover thread EDU per extent reclaim per member
- Incompatible with operations such as backup, load, and others; extent reclaim will yield to other operations and pause automatically

Fast Pre-Allocation

Fast Pre-allocation

Allows the fast preallocation file system feature to reserve space for table space containers, and speed up the process of creating or altering large table spaces and database restore operations.

Supported on the following file systems:

- Veritas VxFS (AIX and Linux),
- GPFS (AIX and Linux),
- JFS2 (AIX only),
- ext4 (Linux only), and
- xFS (Linux only)

Enabled when DB2_USE_FAST_PREALLOCATION is set to ON. Defaults to ON for the combinations above, OFF otherwise

This speed improvement is implemented at a small delta cost of performing actual space allocation during runtime when pages are written to table space containers.

Fast Pre-allocation

Pitfalls

Depending on your operating system, file system type, storage configuration and workload, there may be a noticeable negative performance impact to normal runtime or maintenance operations when this feature is enabled

- Slow restore from backup on AIX JFS2 when fast preallocation is enabled
- Unexpected EOF on RHEL xFS caused by sparse files when fast preallocation is enabled

GPFS with fast preallocation has experienced performance issues during high-volume inserts

- Preallocated regions are created sparse (i.e., holes in the file)
- When DIO is used, GPFS must convert sparse regions to fully allocated blocks before I/O can proceed
- Causes delays, periodic ETIMEOUT, and performance degradation
- GPFS will revert to buffered I/O, further impacting performance

Remote Storage

Remote Storage

Remote storage is storage that is managed by an external service and is accessed through a non-file system interface

Typical use case:

- You have a large database
- Fast local block storage is 5-10x more expensive than remote object storage
- Remote object storage is infinitely scaled
- So, you put the database (catalogs, temps) on the fast local block storage, but all the user tablespaces are stored on object storage

Remote object storage is high-latency. A local cache is used to mitigate this fact.

Supported remote storage providers

- IBM® Cloud Object Storage
- Amazon Simple Storage Service (S3)
- Other object storage providers using the S3 protocol
- Microsoft Azure Blob Storage (LinuxAMD64 RHEL 9.4 only)

Restrictions for Remote Storage Table Spaces

- Only tables organized by column can be stored in remote table spaces. This includes Materialized Query Tables (MQTs) and Declared Global Temporary Tables (DGTT) that are organized by column
- User-defined indexes cannot be stored in remote tablespaces. Users must explicitly select a non-remote table space for the indexes using the "INDEX IN" clause.
  - The indexes created for both the column organized table and its synopsis table will be located in the non-remote table space
  - the internal index for the Page Map Index (PMI) is an exception, as it is created within the same remote table space as the table
- pureScale and HADR cannot be used with databases with remote storage table spaces

Defining Remote Storage Table Spaces

1

Catalog a Storage Access Alias

2

Define a Remote Storage Group on the Remote Alias

3

Create a Table Space on your Remote Storage Group

Defining Remote Storage Table Spaces

Storage Aliases

The first step is telling Db2 how to access your remote storage provider.

`CATALOG STORAGE ACCESS ALIAS …`

- Defines an alias for accessing remote storage
- At minimum, stores your remote storage vendor, the endpoint, and your key/secret pair
- Optional – can specify defaults for
  - CONTAINER: S3 bucket or Azure Blob Storage container
  - OBJECT: name of the object (file) on the remote storage

Can later be updated using `UPDATE STORAGE ACCESS ALIAS …`

When this alias is intended to be used for a remote storage group, the referenced object storage bucket should not be used for any other purpose other than remote storage groups for the specific instance and database

DB2REMOTE Identifiers

`DB2REMOTE://<alias>/<container>/<object>`

- If a container or object is specified in the DB2REMOTE identifier, then it overrides the corresponding value from the alias.
- If a container or object is specified as the empty string in the DB2REMOTE identifier, then the corresponding value, or values, from the alias is used.
- If a container or object is specified as the empty string in the DB2REMOTE identifier, and no corresponding value, or values, exist in the alias, then the DB2REMOTE identifier is invalid and operations using it fail

Defining Remote Storage Table Spaces

Remote Storage Groups

A remote storage group is a storage group defined on a remote storage path

Create a remote storage group

- Define a storage group on a remote storage path using a DB2REMOTE identifier. Example:
  `CREATE STOGROUP SGOBJSTORE ON DB2REMOTE://IBMDEFAULTREMALIAS//`
- **Remote storage groups should have only a single path**

Altering remote storage group

- Remote storage groups cannot be altered. They must be dropped and recreated

Dropping a remote storage group

- The remote storage group can only be dropped if no table space uses the group

Restrictions:
In Db2WHoC, end users will not be allowed to create or drop remote storage groups. In these environments a single remote storage group “IBMDEFAULTREMALIAS” will be pre-created at deployment time

Defining Remote Storage Table Spaces

Creating Remote Storage Table Spaces

```
CREATE TABLESPACE … USING STOGROUP <remote-storage-group>
```

Default Immutable Properties:

- AUTOMATIC STORAGE
- LARGE
- AUTORESIZE YES
- Single container

Unsupported properties because of the different storage engine:

- [NO] FILESYSTEM CACHING
- INITIALSIZE
- INCREASESIZE

Defining Remote Storage Table Spaces

Altering Remote Storage Table Spaces

```
ALTER TABLESPACE …
```

Unsupported properties because of the different storage engine:

- LOWER HIGH WATER MARK and REDUCE
- INCREASESIZE
- AUTORESIZE
- REBALANCE
- CONVERT TO LARGE
- USING STOGROUP

PD & Monitoring

PD & Monitoring

Db2pd – Table Spaces

```
$ db2pd -db TESTDB -tablespaces

Database Member 0 -- Database TESTDB -- Active -- Up 0 days 00:00:10 -- Date 2025-12-01-17.18.58.126196

Tablespace Configuration:
Address            Id    Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe
0x00007FB63F1AB8A0 2     DMS  Large   4096   32       Yes  32       1     1         Def 1        0

LastConsecPg RSE  Name
31           Yes  USERSPACE1

Tablespace Statistics:
Address            Id    TotalPgs   UsablePgs  UsedPgs    PndFreePgs FreePgs    HWM        Max HWM    State
0x00007FB63F1AB8A0 2     8192       8160       96         0          8064       96         96         0x00000000

MinRecTime NQuiescers PathsDropped TrackmodState
0          0          No           n/a
```

PD & Monitoring

Db2pd – Table Spaces (cont.)

```
Tablespace Autoresize Statistics:
Address            Id    AS  AR  InitSize            IncSize             IIP MaxSize
0x00007FB63F1AB8A0 2     Yes Yes 33554432            -1                  No  None

LastResize                LRF
None                      No

Tablespace Storage Statistics:
Address            Id    DataTag  Rebalance SGID  SourceSGID
0x00007FB63F1AB8A0 2     -1       No        0     -

Containers:
Address            TspId ContainNum Type    TotalPgs   UseablePgs PathID     StripeSet  Container
0x00007FB61EC91860
2     0          File    8192       8160       0          0          /Path1/NODE0000/TESTDB/T0000002/C0
000000.LRG
```

PD & Monitoring

Db2pd – Extent Movement

```
$ db2pd –db TESTDB -extentmovement

Database Member 0 -- Database TESTDB -- Active -- Up 0 days 00:04:33 -- Date 2012-10-26-11.19.52.056414

Extent Movement:
Address            TbspName Current Last Moved Left TotalTime
0x00002AAB356D4BA0 DAVID    1168    1169 33    426  329636
```

PD & Monitoring

Db2pd – Storage Groups and Storage Paths

```
$ db2pd -db testdb -storagegroups

  Storage Group Configuration:
  Address            SGID  Deflt DataTag  Name
  0x00002BA9E6CFF4C0 0    Yes   0       IBMSTOGROUP

  Storage Group Statistics:
  Address            SGID State      NumPaths  NumDropPen
  0x00002BA9E6CFF4C0 0    0x00000000   1        0

  Storage Group paths:
  Address            SGID PathID  PathState  PathName
  0x00002BA99CD23540 0    0      InUse     /filesystem1
```

PD & Monitoring

MON_GET_EXTENT_MOVEMENT_STATUS() - extent movement status for one or more table spaces

```
$ db2 SELECT * FROM TABLE(SYSPROC.MON_GET_EXTENT_MOVEMENT_STATUS('TS_01A', -1))

TBSP_NAME      TBSP_ID  MEMBER CURRENT_EXTENT LAST_EXTENT NUM_EXTENTS_MOVED
-------------- -------- ------ -------------- ----------- -----------------
TS_01A                3      0             -1          -1                -1

...NUM_EXTENTS_LEFT TOTAL_MOVE_TIME   PHYSICAL_MOVE_TIME   LBP_UPDATE_TIME
...---------------- ----------------- -------------------- -----------------
...               -1                -1                   -1                -1

...REMOTE_UPDATE_TIME   GBP_UPDATE_TIME   TOTAL_METADATA_UPDATE_TIME EHL_LOCKWAIT_TIME
...-------------------- ----------------- ------------------------- --------------------
...                  -1                -1                        -1                   -1

  5 record(s) selected.
```

PD & Monitoring

MON_GET_CONTAINER() - monitor metrics for one or more table space containers

```
$ db2 SELECT * FROM TABLE(MON_GET_CONTAINER('',-2))
```

- Direct reads + writes and time spent doing direct reads + writes
- Pages read + written
- Vectored + block I/Os
- File system total size and file system used size
- Caching tier direct reads, number of caching tier direct read requests, and time spent
- … and more

PD & Monitoring

MON_GET_TABLESPACE() - monitor metrics for one or more table spaces

```
$ db2 SELECT * FROM TABLE(MON_GET_TABLESPACE('',-2))
```

- Table space type (DMS or SMS)
- Content (ANY, LARGE, SYSTMP, USRTMP)
- Page, extent, prefetch size
- Rebalancer mode
- Logical, physical, and asynchronous reads for data, index, XML (permanent, temporary, local buffer pool, global buffer pool)
  - All the same for caching tier as well
- Max size
- Increase size
- Remote storage write bytes, write requests, and write time
- … and more

PD & Monitoring

MON_GET_TABLESPACE_QUIESCER() - information about quiesced table spaces

```
$ db2 SELECT * FROM TABLE(MON_GET_TABLESPACE_QUIESCER(-2))

TBSP_NAME  QUIESCER_TS_ID  QUIESCER_OBJ_ID  QUIESCER_AUTH_ID  ...
---------- -------------- ---------------- ----------------- ...
USERSPACE1               2                5 SWALKTY           ...
USERSPACE1               2                5 SWALKTY           ...

... QUIESCER_APPLICATION_HANDLE QUIESCER_STATE DBPARTITIONNUM
... -------------------------- -------------- --------------
...                          0 EXCLUSIVE                    0
...                      65983 EXCLUSIVE                    1

  2 record(s) selected.
```

PD & Monitoring

MON_GET_TABLESPACE_RANGE() - displays information about table space ranges

```
$ db2 SELECT * FROM TABLE(MON_GET_TABLESPACE_RANGE(-2))
```

- Range number
- Range offset
- Range max page and max extent
- Range start and end stripe
- Range adjustment
- Number of containers in range
- Range container ID

PD & Monitoring

MON_GET_REBALANCE_STATUS() - get rebalance progress for a table space

```
$ db2 SELECT * FROM TABLE(MON_GET_REBALANCE_STATUS(NULL,-2))

TBSP_NAME                      DBPARTITIONNUM MEMBER REBALANCER_MODE
------------------------------ -------------- ------ ------------------------------
SYSCATSPACE                                 0      0 REV_REBAL

REBALANCER_STATUS REBALANCER_EXTENTS_REMAINING REBALANCER_EXTENTS_PROCESSED
----------------- ---------------------------- ----------------------------
ACTIVE                                   6517                            4

REBALANCER_START_TIME
--------------------------
2011-12-01-12.08.16.000000

  1 record(s) selected.
```

Best Practices

Best Practices

- Use dedicated storage for your table spaces. Separate transaction logs from table space data
- Create your file systems uniformly and of the same size
- Distribute your table spaces evenly across file systems
- Use automatic storage
- Migrate non-reclaimable table spaces to reclaimable storage automatic DMS
- Put LONG VARCHAR / LOB data in dedicated tablespaces with filesystem caching enabled
- Plan REBALANCE and REDUCE operations for low traffic or maintenance windows
- Avoid keeping just one table per tablespace. Database activation time linearly increases with the number of tablespaces

References

- www.ibm.com/docs/en/db2/12.1.x?topic=commands-catalog-storage-access
- www.ibm.com/docs/en/db2/12.1.x?topic=data-page-table-table-space-size
- www.ibm.com/docs/en/db2/12.1.x?topic=groups-default-storage
- www.ibm.com/docs/en/db2/12.1.x?topic=management-database-managed-space
- www.ibm.com/docs/en/db2/12.1.x?topic=management-reclaimable-storage
- www.ibm.com/docs/en/db2/12.1.x?topic=parameters-multipartsizemb-remote-storage-multipart-upload-part-size
- www.ibm.com/docs/en/db2/12.1.x?topic=spaces-storage-group-table-space-media-attributes
- www.ibm.com/docs/en/db2/12.1.x?topic=spaces-table-system-user-temporary-data
- www.ibm.com/docs/en/db2/12.1.x?topic=statements-create-tablespace
- www.ibm.com/docs/en/db2/12.1.x?topic=statements-create-table
- www.ibm.com/docs/en/db2/12.1.x?topic=storage-db2remote-identifiers
- www.ibm.com/docs/en/db2/12.1.x?topic=storage-remote-requirements
- www.ibm.com/docs/en/db2/12.1.x?topic=tools-systoolspace-systoolstmpspace-table-spaces

References

- www.ibm.com/docs/en/db2/12.1.x?topic=variables-miscellaneous#r0005669__M_DB2_ENABLE_COS_SDK
- www.ibm.com/docs/en/db2/12.1.x?topic=variables-performance
- www.ibm.com/docs/en/db2-warehouse?topic=support-managing-remote-table-spaces
- www.ibm.com/docs/en/storage-scale/5.2.3?topic=applications-considerations-use-direct-io-o-direct
- www.ibm.com/support/pages/db2-backup-fails-adm6006e-db2-v11-running-linux-redhat-using-xfs-file-system
- www.ibm.com/support/pages/effect-db2usefastpreallocation-db2-restore
- www.ibm.com/support/pages/how-troubleshoot-and-fix-slow-db2-restore-aix-jfs2-file-system
- www.ibm.com/docs/en/db2/12.1.x?topic=mmr-mon-get-container-get-table-space-container-metrics
- www.ibm.com/docs/en/db2/12.1.x?topic=mmr-mon-get-extent-movement-status-get-extent-movement-progress
- www.ibm.com/docs/en/db2/12.1.x?topic=routines-mon-get-tablespace-get-table-space-metrics
- www.ibm.com/docs/en/db2/12.1.x?topic=mmr-mon-get-rebalance-status-table-function-get-rebalance-progress-table-space

References

- www.ibm.com/docs/en/db2/12.1.x?topic=mmr-mon-get-tablespace-quiescer-get-information-about-quiesced-table-spaces
- www.ibm.com/docs/en/db2/12.1.x?topic=mmr-mon-get-tablespace-range-get-information-about-table-space-ranges
- www.ibm.com/docs/en/db2/12.1.x?topic=sql-xml-limits
- www.ibm.com/docs/en/db2/12.1.x?topic=commands-db2pd-monitor-troubleshoot-db2-engine-activities
- www.ibm.com/docs/en/db2/12.1.x?topic=statements-create-stogroup

Appendix

Automatic Storage: DMS
Naming Convention

Catalogs:
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######**.CAT**

System Temporary Tablespaces:
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######**.TMP**

User Temporary Tablespaces:
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######**.UTM**

Regular Tablespaces:
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######**.USR**

Large Tablespaces:
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######**.LRG**

Automatic Storage: SMS
Naming Convention

System Temporary Tablespaces:

```
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######.TMP/*
```

User Temporary Tablespaces:

```
<storage path>/<instance>/NODE####/<dbname>/T#######/C#######.UTM/*
```

Automatic Storage: SMS
Naming Convention

Container tag: `SQLTAG.NAM`

Files for database objects take the form of:

`SQL<object-id>.<type-extension>`

- Object ID is padded to four digits
- In pureScale, *".MEMBERxxxx"* is appended to SMS temps
- Each database object (table, index, …) resides in a separate file
- Mapping ***Object ID*** to the object name:
  `SELECT NAME FROM SYSIBM.SYSTABLES WHERE FID = :objectID and TID = :tsID`

Automatic Storage: SMS Naming Convention

Object Type Extensions:

EXT	MEANING
.DAT	Data
.TDA	Temporary Data
.DTR	Reorg Temporary Data
.INX	Index
.TIX	Temporary Index
.IN1	Shadow Index

EXT	MEANING
.LF	Long Field (LF)
.TLF	Temporary LF
.LFR	Reorg Temporary LF
.LB	Large Object (LOB)
.TLB	Temporary LOB
.RLB	Reorg Temporary LOB
.LBA	Lob Allocation (LBA)
.TBA	Temporary LBA
.RBA	Reorg Temporary LBA

EXT	MEANING
.BKM	Block Map (BMP)
.TBM	Temporary BMP
.BMR	Reorg Temporary BMP
.XDA	XML Storage (XDA)
.TXD	Temporary XDA
.RXD	Reorg Temporary XDA
.CDE	Columnar Data Engine
.TCE	Temporary CDE