An Introduction to Automatic Storage

CODUG – May 19th, 2015

Kelly Schlamb
Executive IT Specialist, Worldwide Analytics Platform Technical Sales
IBM Canada Ltd.

Email: *kschlamb@ca.ibm.com*   *Twitter:* *@KSchlamb*

Disclaimer

Agenda

- **Overview of Automatic Storage and Feature Highlights**

- **Multi-Temperature Data and Storage Management**

- **Moving To Automatic Storage**

- **Best Practices**

Automatic Storage

- **Single point of storage management for table spaces**
  - Introduced in DB2 8.2 FP9
  - Default for newly created databases as of DB2 9.1

- **A set of storage paths provide a pool of storage to the database**
  - Multiple pools available via storage groups in DB2 10.1

- **Easy creation of table spaces that are automatically striped across all of the storage paths**
  - Table spaces grow from the storage pool on demand

Non-Automatic Storage
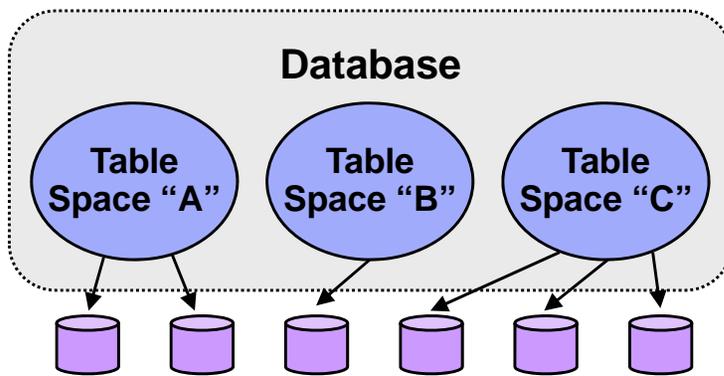
Automatic Storage
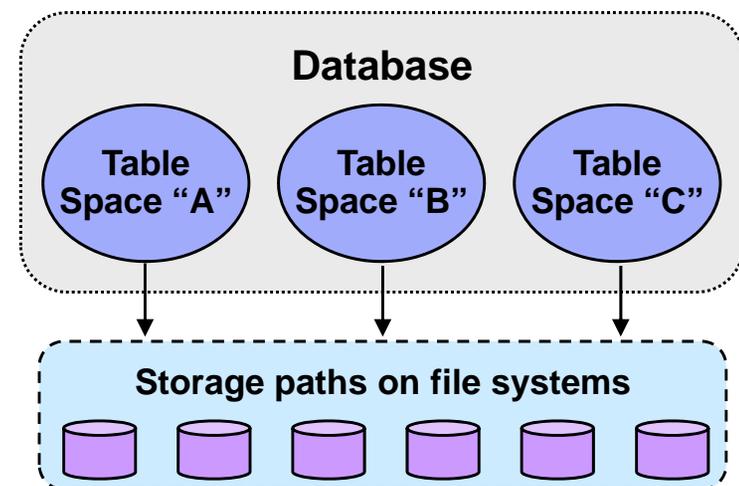
Feature Highlights by Release

- **DB2 8.2 FP9**
  - Automatic storage (AS) introduced
  - Auto-resize DMS table spaces

- **DB2 9.1**
  - AS default for newly created databases
  - DPF support for AS

- **DB2 9.5**
  - Ability to reduce size of AS table spaces

- **DB2 9.7**
  - Reclaimable storage
  - Migration to AS
  - Drop storage paths
  - Rebalance AS table spaces

- **DB2 10.1**
  - Storage Groups
  - Suspend rebalance

Automatic Storage with DPF

- **The same list of storage paths are used on each partition**
  - However, underlying file systems do not have to have the same space characteristics
    - Depending on layout, some partitions may be expected to hold more data than others
    - Each partition will act independently when it comes to growth

- **For cases where a storage path is shared by database partitions (e.g. multiple logical partitions) it is suggested that individual mount points be created for each partition:**

  e.g. `/dbdata/dbinst/`**`NODE0000`** <= Mounted file system #1
  `/dbdata/dbinst/`**`NODE0001`** <= Mounted file system #2

- **Alternatively, database partition expressions can be used within the storage paths**

  e.g. Storage path is specified as: `/dbdata/dir` **`$N`**

  Storage path actually used on partition 0: `/dbdata/dir`**`0`**
  Storage path actually used on partition 1: `/dbdata/dir`**`1`**
  …

Reclaimable Storage

- **An attribute of new DMS and AS table spaces created in DB2 9.7 or later**

```
SELECT TBSP_NAME, RECLAIMABLE_SPACE_ENABLED FROM
    TABLE(MON_GET_TABLESPACE(NULL,NULL))
```

- **Allows you to free up unused extents below the high-water mark**

- **Cannot migrate older table spaces to this new storage model – consider using the online table move procedure instead**

  1. Create new version of table space (automatically has reclaimable storage capability)
  2. Use `ADMIN_MOVE_TABLE` procedure to move tables from old table space to new table space
  3. Drop old version of table space
  4. Rename new version of table space to have the old name (if desired)

Reclaimable Storage (cont.)

```
ALTER TABLESPACE <tsName> REDUCE --+-------------------------------+---
                                   '-- <size> --+-------------+--+--'
                                   |            +----- K -----+  |
                                   |            +----- M -----+  |
                                   |            +----- G -----+  |
                                   |            '-- PERCENT --'  |
                                   '--- MAX ----------------------'
                                   '--- STOP ---------------------'
```

- **Existing `REDUCE` option enhanced to free "trapped" free space**

- **Works by**
  - Moving used extents from higher addresses in the table space to unused lower addresses
  - Lowering high water mark accordingly
  - Shrinking/removing containers to return space back to the automatic storage paths

- **`MAX` frees as much space as possible (can also specify a target size to free up)**

- **Runs in the background**
  - Works in batches, committing free extents as it progresses
  - `STOP` option terminates a background `REDUCE` operation

Reclaimable Storage: Example

DROP TABLE T2
DROP TABLE T3

ALTER
TABLESPACE …
REDUCE MAX
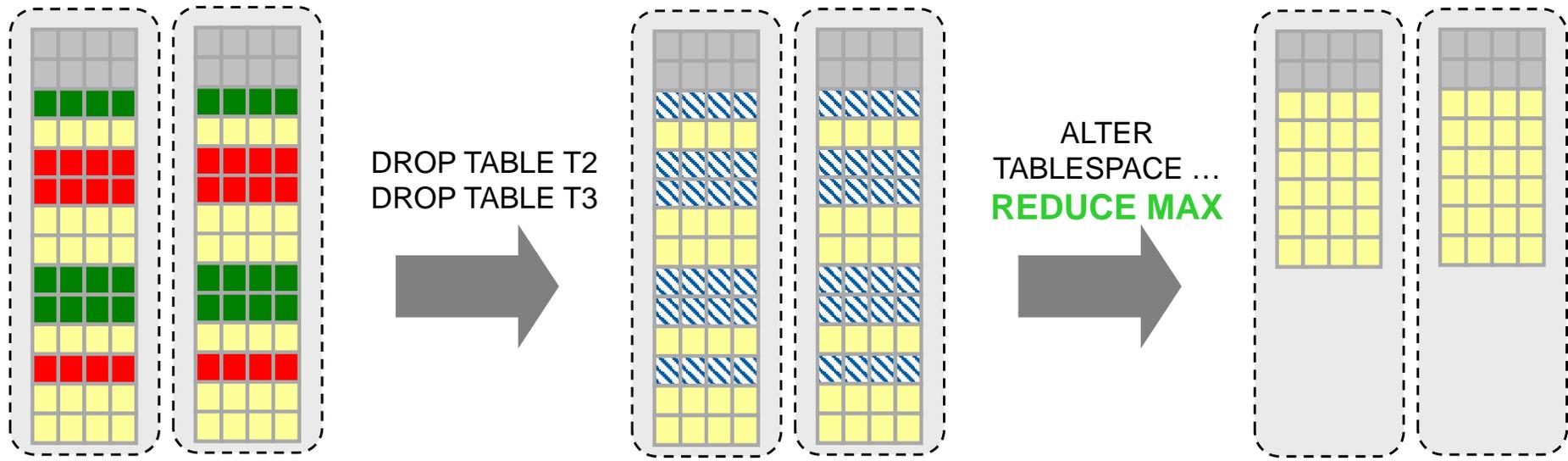
Internal table space metadata extents

Table T1

Table T2

Table T3

Extent that is allocated to table space, but not to a table

Storage Groups

- **Introduced in DB2 10.1**

- **Provides layer of abstraction between logical storage (table spaces) and physical storage (containers)**

- **A storage group is a named set of storage paths where table space data can be stored**

- **Adds flexibility to automatic storage to allow for more than one storage pool within a database**
  – Can represent different classes of storage (slow vs. fast HDDs, SSDs)
  – Useful for multi-temperature data management (more on this later)

- **1:N mapping of storage groups to automatic storage table spaces**
  – Can move a table space from one storage group to another online

Database

Table Space "A" Table Space "B" Table Space "C"

SG1 SG2
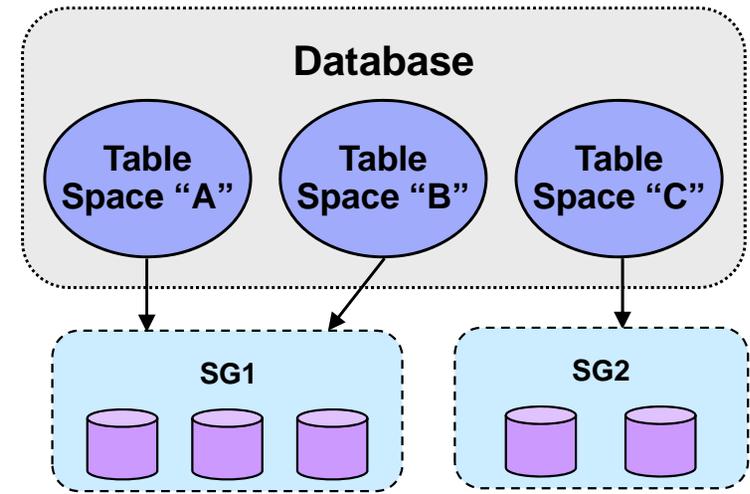
Storage Groups (cont.)

- **New DDL to support storage group management**
  - CREATE STOGROUP
  - ALTER STOGROUP
  - DROP STOGROUP
  - RENAME STOGROUP

- **Media attributes can be set for storage groups**
  - Overhead: Device seek and latency time, in milliseconds
  - Device Read Rate: Read transfer rate, in megabytes per second
  - Data Tag: Tag used by table spaces in this storage group (more later)

- **Can add storage paths and drop storage paths from a storage group**

- **IBMSTOGROUP is default storage group**
  - Can specify any storage group as the default

Storage Groups (cont.)

- **Storage group specified when AS table space created**
  - CREATE TABLESPACE … USING STOGROUP <SGName>
  - Default storage group is used if one not specified
    - Any storage group can be designated as the default

- **Table space can move from one storage group to another**
  - ALTER TABLESPACE … USING STOGROUP <SGName>
  - Commonly used for change of temperature when implementing multi-temperature data management
  - Implicitly involves a rebalance from containers in current storage group to target storage group
    - Online and no loss of data availability
    - Monitor progress via MON_GET_REBALANCE_STATUS

- **Table spaces inherit media attributes of the storage group by default**
  - Overhead, device read rate, data tag
  - Can override these attributes at table space level if desired

Storage Groups and Table Spaces – Example

`CREATE STOGROUP SG1 ON '/sg1-p1'`

Create new storage group SG1

`CREATE TABLESPACE TS1 USING STOGROUP SG1`

Create table space TS1 within storage group SG1

`ALTER STOGROUP SG1 ADD '/sg1-p2'`

Add a new storage path to storage group SG1

`ALTER TABLESPACE TS1 REBALANCE`

Rebalance table space TS1 across the newly added storage path in SG1

`CREATE STOGROUP SG2 ON '/sg2-p1', '/sg2-p2'`

Create new storage group SG2

`ALTER TABLESPACE TS1 USING STOGROUP SG2`

Move table space TS1 from SG1 to SG2 – implicitly rebalances from SG1's paths to SG2's paths

Monitoring Storage Groups and Usage

- **`ADMIN_GET_STORAGE_PATHS` table function**
  - Returns a list of storage paths for each database storage group, including file system information for each path

```
SELECT CHAR(STORAGE_GROUP_NAME, 10) AS SGNAME,
       CHAR(DB_STORAGE_PATH, 15) AS PATH,
       CHAR (DB_STORAGE_PATH_STATE, 10) AS STATE,
       INT(FS_TOTAL_SIZE / (1024*1024)) AS TOTAL_MB,
       INT(FS_USED_SIZE / (1024*1024)) AS USED_MB,
       INT(STO_PATH_FREE_SIZE / (1024*1024)) AS FREE_MB
  FROM TABLE(ADMIN_GET_STORAGE_PATHS('', -1))
  WHERE STORAGE_GROUP_NAME = 'SG1'
```

SGNAME	PATH	STATE	TOTAL_MB	USED_MB	FREE_MB
SG1	/sg1path	IN_USE	738514	585567	50961

Monitoring Storage Groups and Usage (cont.)

- **`MON_GET_TABLESPACE` table function**
  - Returns list of table spaces along with storage group information

```
SELECT CHAR(TBSP_NAME, 15) AS NAME,
       TBSP_ID AS ID,
       CHAR(TBSP_CONTENT_TYPE, 10) AS TYPE,
       TBSP_USING_AUTO_STORAGE AS AUTOSTG,
       CHAR(STORAGE_GROUP_NAME, 13) AS SGNAME
  FROM TABLE(MON_GET_TABLESPACE('', -1)) ORDER BY TBSP_ID

NAME                 ID        TYPE       AUTOSTG SGNAME
----------------- -------- ---------- ------- --------------
SYSCATSPACE               0 ANY              1 IBMSTOGROUP
TEMPSPACE1                1 SYSTEMP          1 IBMSTOGROUP
USERSPACE1                2 LARGE            1 IBMSTOGROUP
TS1                      3 LARGE            1 SG1
```

Agenda

- **Overview of Automatic Storage and Feature Highlights**

- **Multi-Temperature Data and Storage Management**

- **Moving To Automatic Storage**

- **Best Practices**

Why Multi-Temperature Data Management?
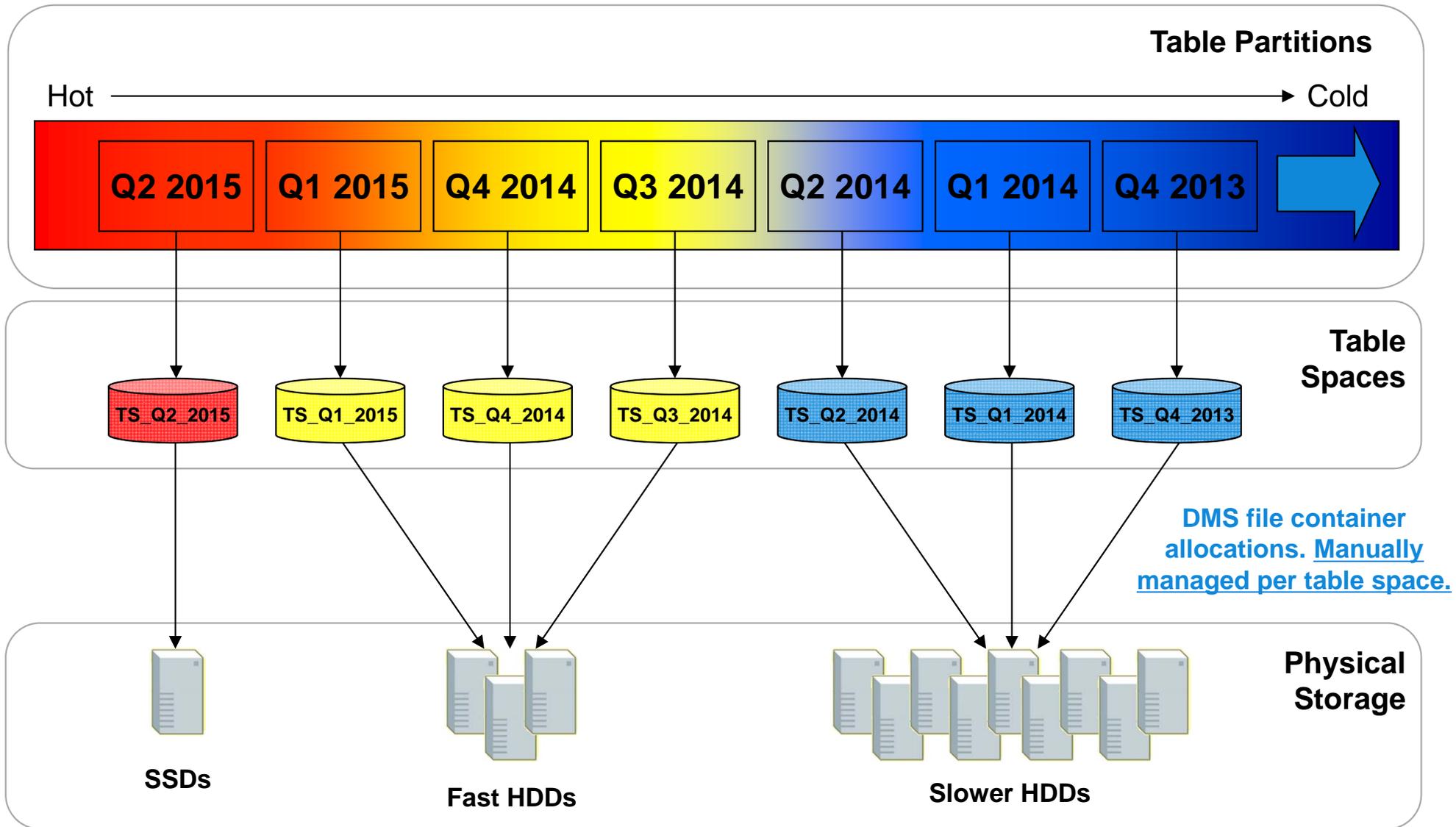
- **Data warehouses retain large volumes of data, which can be classified according to its "temperature"**

- **Hotter data**
  - New
  - Frequently accessed
  - Smaller proportion of warehouse
  - Queries are expected to perform optimally

- **Colder data**
  - Aged
  - Accessed infrequently
  - Larger proportion of warehouse
  - Query performance not as critical

Data Volume / Age

Usage

HOT

WARM

COLD

DORMANT

- **Using faster, more expensive storage devices for hot data and slower, less expensive storage devices for cold data optimizes the performance of the queries that matter most while helping to reduce overall cost**
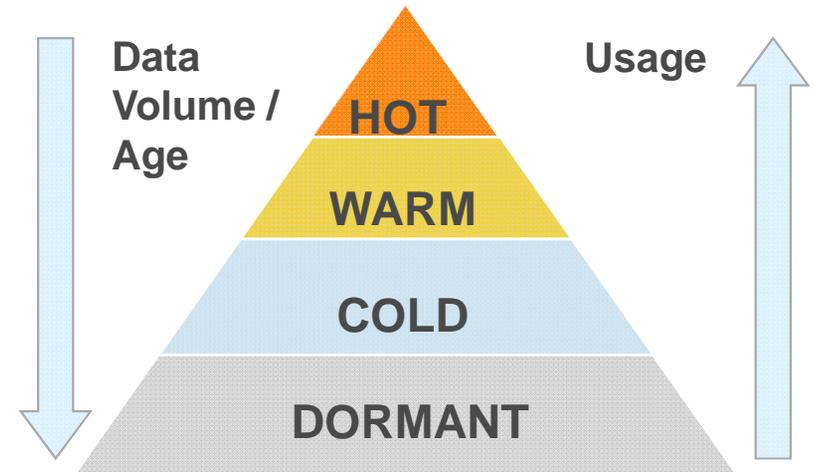
Multi-Temperature Storage (pre-DB2 10.1)

- **Automatic storage not typically a good fit as there is only a single pool of storage for the database**
  - Cannot place table spaces onto different classes of storage

- **SMS table spaces should be avoided**
  - Physical container locations cannot be moved easily

- **Use DMS table spaces**
  - Containers can be added or dropped while table space is online
    - Add containers on target storage
    - Drop containers from original storage
  - Implicit rebalance moves data after each step of add and drop

Multi-Temperature Storage (pre-DB2 10.1) (cont.)

Table Partitions

Hot ──→ Cold

| Q2 2015 | Q1 2015 | Q4 2014 | Q3 2014 | Q2 2014 | Q1 2014 | Q4 2013 |

Table Spaces

TS_Q2_2015   TS_Q1_2015   TS_Q4_2014   TS_Q3_2014   TS_Q2_2014   TS_Q1_2014   TS_Q4_2013

DMS file container allocations. Manually managed per table space.

Physical Storage

SSDs

Fast HDDs

Slower HDDs

Multi-Temperature Storage (pre-DB2 10.1) (cont.)

- **Create "hot" table space on fast SSD storage**

  - ```
    CREATE TABLESPACE TS_Q2_2015 MANAGED BY DATABASE
        USING (FILE '/ssdpath1/q2_2015_c1' 200M,
               FILE '/ssdpath2/q2_2015_c2' 200M)
        OVERHEAD 4 TRANSFERRATE 0.07
    ```

- **After some period of time, table space needs to be moved to slower "warm" HDD storage**

  - ```
    ALTER TABLESPACE TS_Q2_2015
        BEGIN NEW STRIPE SET
            (FILE '/hddpath1/q2_2015_c1' 200M,
             FILE '/hddpath2/q2_2015_c2' 200M)
        OVERHEAD 8 TRANSFERRATE 0.2
    ```
    No need for rebalance of data at this point

  - ```
    ALTER TABLESPACE Q2_2015_DATA
        DROP (FILE '/ssdpath1/q2_2015_c1',
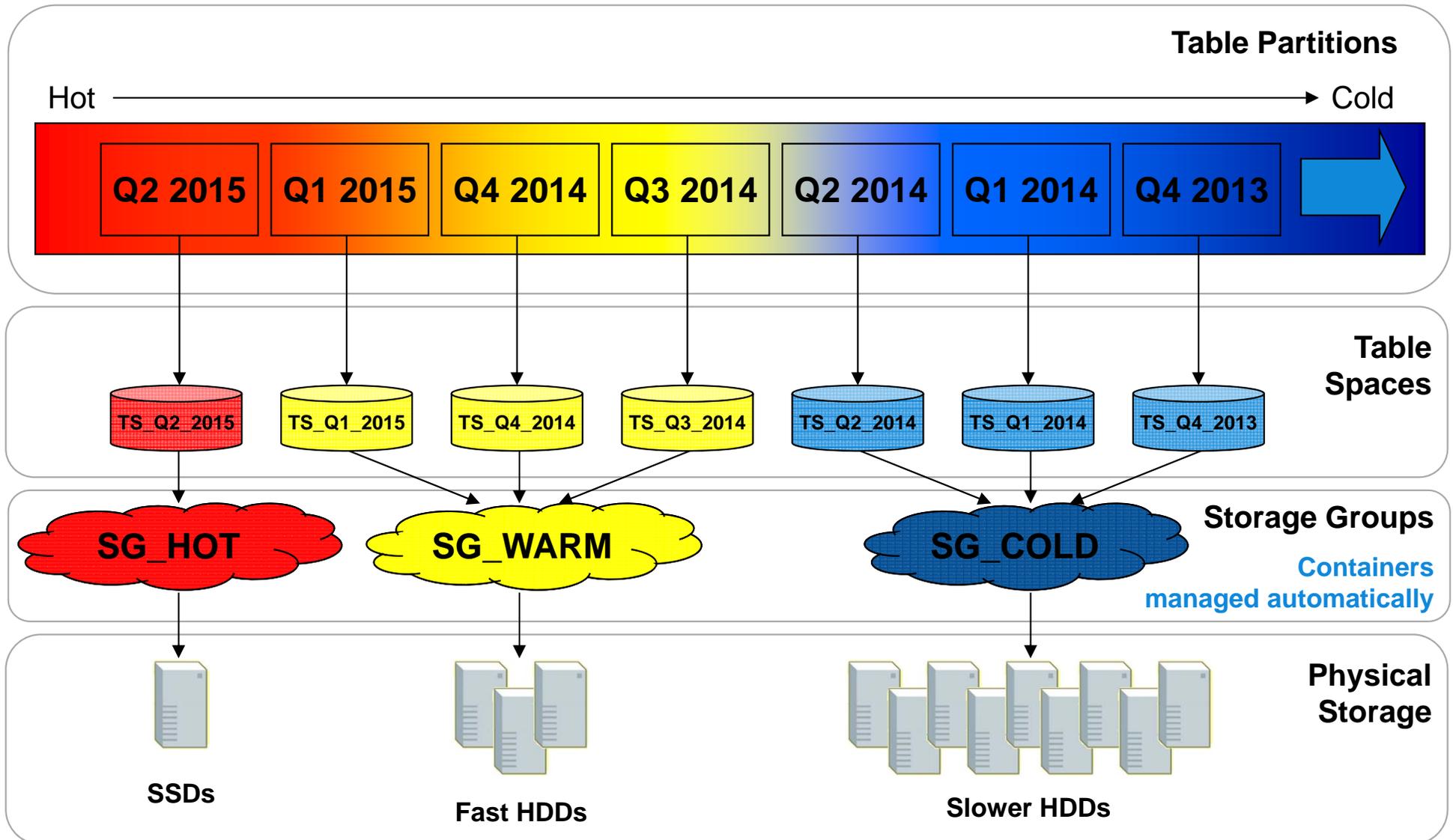   FILE '/ssdpath2/q2_2015_c2')
    ```
    Data rebalanced off of old containers

# Multi-Temperature Storage (DB2 10.1)

- **Use storage groups to create pools of storage**
  - e.g. hot, warm, cold

- **Associate table spaces with the storage groups**
  - Table spaces dynamically inherit media attributes of storage group

- **Change table space -> storage group assignments over time as business needs dictate**
  - Online movement of data with no loss of availability

# Multi-Temperature Storage (DB2 10.1) (cont.)

**Table Partitions**

Hot ⟶ Cold

| Q2 2015 | Q1 2015 | Q4 2014 | Q3 2014 | Q2 2014 | Q1 2014 | Q4 2013 |

**Table Spaces**

TS_Q2_2015   TS_Q1_2015   TS_Q4_2014   TS_Q3_2014   TS_Q2_2014   TS_Q1_2014   TS_Q4_2013

**Storage Groups**

**SG_HOT**   **SG_WARM**   **SG_COLD**

*Containers managed automatically*

**Physical Storage**

SSDs   Fast HDDs   Slower HDDs

# Multi-Temperature Storage (DB2 10.1) (cont.)

- **Create storage groups**
  - CREATE STOGROUP **SG_HOT** ON **'/ssdpath1', '/ssdpath2'**
    DEVICE READ RATE 300 OVERHEAD 2.5 DATATAG 1
  - CREATE STOGROUP **SG_WARM** ON **'/hddpath1', '/hddpath2'**
    DEVICE READ RATE 100 OVERHEAD 6 DATATAG 5

- **Create table space and assign storage group**
  - CREATE TABLESPACE TS_Q2_2015
    USING STOGROUP **SG_HOT**

- **Change storage group**
  - ALTER TABLESPACE TS_Q2_2015
    USING STOGROUP **SG_WARM**

**Data rebalanced off of storage paths for SG_HOT and onto storage paths for SG_WARM**

**Data movement is online!**

**No loss of data availability!**

# Multi-Temperature Data Management (DB2 10.1)

- **Improved performance through tight integration with DB2 Workload Management**
  - Data tags signify business priority
  - Proactively/reactively map activities based on data tag and threshold values

- **Optim Configuration Manager automates movement for range partitioned tables, based on age of data**

- **Improve manageability of storage while saving on storage cost and satisfying performance requirements**

# OCM: Automate Data Aging Across Storage Tiers

- **With Optim Configuration Manager you can define a data migration job that consists of several data migration aging policies for one, some or all of your partitioned tables**

**Hot Storage Group** — Aug | Sep | Oct

**Warm Storage Group** — May | Jun | Jul

**Cold Storage Group** — Nov | ... | Mar | Apr

# OCM: Defining a Data Migration Job

# OCM: Data Migration Aging Policies

- **Defining a data migration aging policy**



- **Displaying a data migration aging policy**

# Integration with DB2 Workload Manager

- **Previously, WLM gave users the ability to prioritize their work based on**
  - "Who" submitted the work
  - "What" type of work it is

- **DB2 10.1 introduces a data-centric ("where") approach**
  - Priority may be given to requests based on where the data being accessed resides

- **A data tag can be assigned to storage groups and table spaces**
  - Valid values are in the range of 0 to 9 (0 means no tag)
  - A higher value represents a higher priority
  - A table space inherits the data tag from its storage group by default
    - Can be overridden

# Integration with DB2 Workload Manager (cont.)

- **DB2 WLM work class and threshold DDL have been extended to support the new data tag attribute**

- **Work class sets are predictive based on compilation information**
  - Sometimes there isn't enough information at compile time (for example, queries with parameter markers) to predict which table spaces will be touched
  - For this reason, activities can be mapped to a service subclass based on the data expected to be touched before the activity starts to run

- **Data tag thresholds are reactive and use information that is available at runtime**
  - Activities can be remapped to a service subclass based on the data touched by the activity as it is running
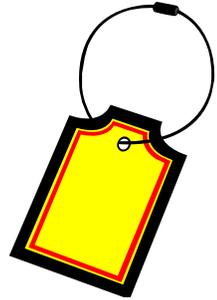
# Workload Management – Predictive Mapping

- **Map statement to service class (predictive)**
- **SQL compiler predicts what data (and table spaces) will be touched by the SQL statement and builds the list of data tags**
- **List of data tags will define the initial service class placement**

Superclass `MAINSC`
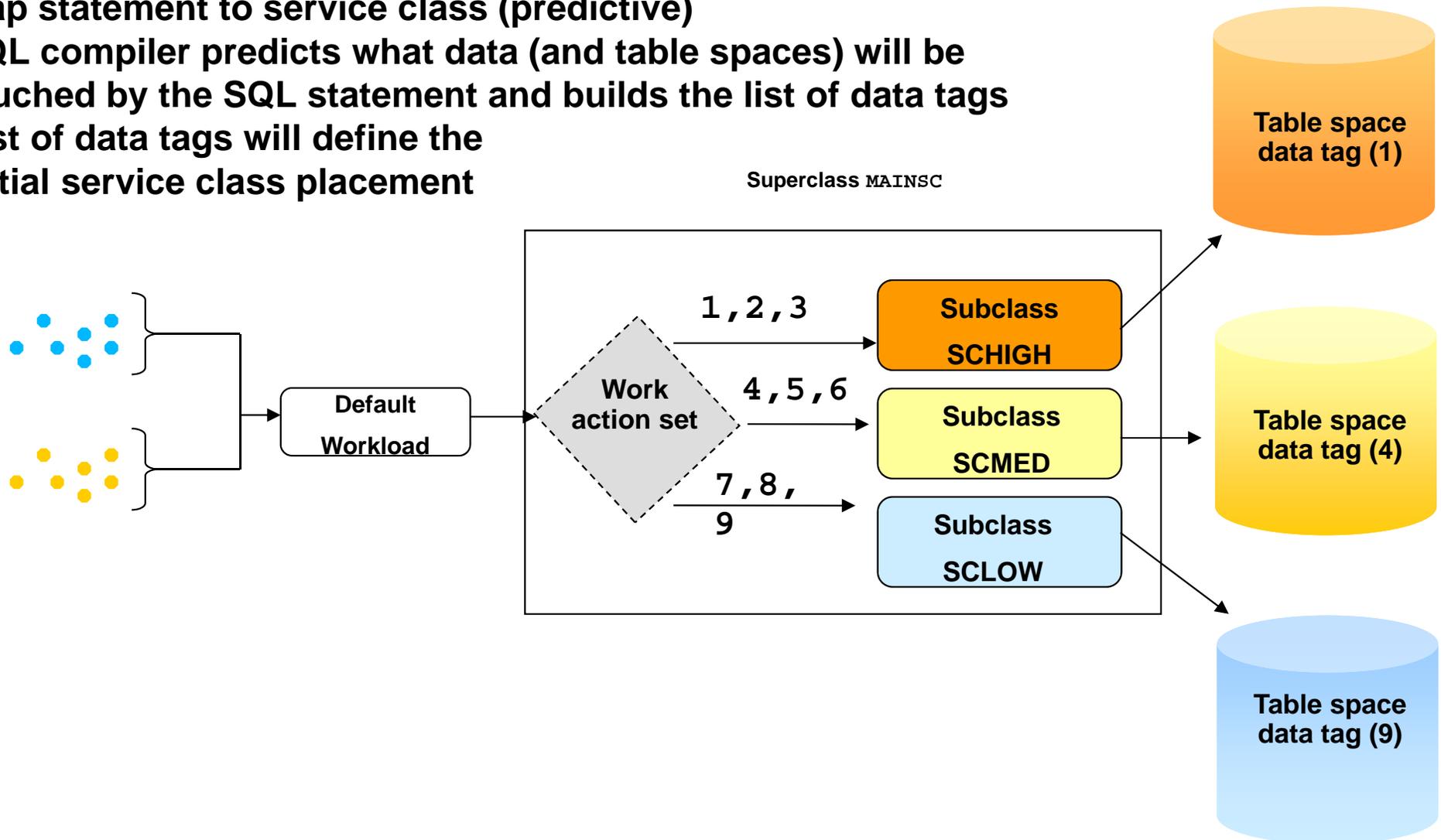
# Workload Management – Reactive Mapping

- **Allows changing priority of workload at runtime based on data accessed (reactive)**
  - New data tag threshold with remap action
  - At runtime, based on tag of table space accessed, place statement in a different service class

Superclass `MAINSC`

Default Workload → Work action set → Subclass SCHIGH / Subclass SCMED / Subclass SCLOW

Table space data tag (1)

Table space data tag (4)

Table space data tag (9)

# Agenda

- **Overview of Automatic Storage and Feature Highlights**

- **Multi-Temperature Data and Storage Management**

- **Moving To Automatic Storage**

- **Best Practices**

# Moving to Automatic Storage

- **Enable database for automatic storage**
  - **DB2 9.7:** Add storage paths to the database
    - `ALTER DATABASE ADD STORAGE …`
    - Creates the default storage group for the database (`IBMSTOGROUP`)
  - **DB2 10.1:** Preferred approach is to explicitly create a storage group
    - Becomes the default for the database
  - Existing table spaces are not impacted at this point

- **Drop temporary table spaces and recreate as AS**

- **Can convert DMS table spaces to AS using one of**
  1. Altering the table space (fully online)
     - `ALTER TABLESPACE … MANAGED BY AUTOMATIC STORAGE`
       - **DB2 10.1:** Can choose storage group to be in
     - `ALTER TABLESPACE … REBALANCE`
  2. Redirected restore of table spaces (backup can be online, restore offline)
     - `SET TABLESPACE CONTAINERS … USING AUTOMATIC STORAGE`
     - **DB2 10.1:** Default storage group is used

# Moving to Automatic Storage (cont.)

- **SMS cannot be converted to DMS or AS**
  - Fundamentally different architectures
  - Must rebuild the table space (e.g. `ADMIN_MOVE_TABLE`)

- **Other things to consider:**
  - Do you need to use the reclaimable storage model added in DB2 9.7?
  - Do you need to convert SYSCATSPACE from SMS to AS?

- **See developerWorks article for more details**
  - `http://www.ibm.com/developerworks/data/bestpractices/purescaleupgrade`

# Database and Table Space Conversion Example

DMS table space TS1
has single container
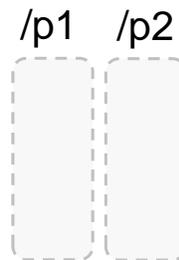
Stripe set of containers on SG1's
storage paths is created. No data
movement takes place

**SG1**

/cont1

**CREATE STOGROUP SG1
ON '/p1', '/p2'**

/cont1    /p1    /p2

**ALTER TABLESPACE TS1
MANAGED BY AUTOMATIC
STORAGE USING STOGROUP SG1**

/cont1    /p1    /p2
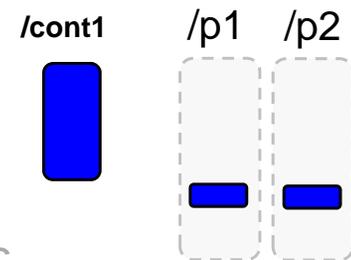
Rebalance moves extents to the new containers
(extending them as necessary to accommodate the
move) and the old container is subsequently removed

**Optionally,** If reclaimable storage is supported and
there is significant free space below the high-water mark
then reduce the table space to free it up

**ALTER TABLESPACE TS1
REBALANCE**

/p1    /p2

**ALTER TABLESPACE TS1
REDUCE MAX**

/p1    /p2
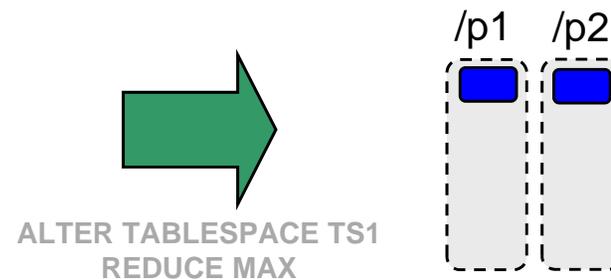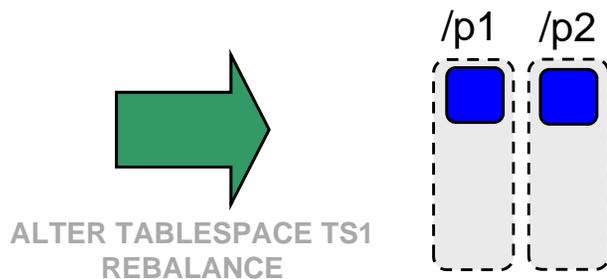
# Handling SYSCATSPACE

- **Functionally, no issue with SYSCATSPACE remaining as SMS**
  - SMS was default in DB2 8.2 and earlier
  - AS is default in DB2 9.1 and later

- **However, if SYSCATSPACE is SMS and there is a desire or need to move to AS then two options exist:**
  - Rebuild the database (could use `db2look` and `db2move`)
  - Might be able to use Transportable Schema feature to move all existing table spaces into a new database
    - Target database created as AS, which means that SYSCATSPACE will be AS
    - See Information Center for restrictions on objects that can't be transported
    - Not supported for DPF configurations

- **If DB2 pureScale is in your future then note that all table spaces – including SYSCATSPACE – must be AS**

# Transportable Schemas

- **`TRANSPORT` feature of database restore introduced in DB2 9.7 FP2**

- **Using a backup image as the source, allows you to copy a set of table spaces and SQL schemas from one database to another**
  - A temporary staging database is created as part of the transport
  - DDL associated with the schemas is captured and replayed on the target
  - Can use `SET TABLESPACE CONTAINERS` as part of a redirected transport restore to change the containers
    - Can convert the remaining table spaces from DMS to AS at this point

- **Transportable Schemas cannot be used for DPF configurations**

# Transportable Schemas (cont.)

**The following logical objects are re-created:**

- Tables, CGTTs, MQTS, and normal and statistical views
- The following types of generated columns:
  - Expression
  - Identity
  - Row change timestamp
  - Row change token
- User-defined functions, user-defined types, and generated functions
- The following types of constraints:
  - Check
  - Foreign key
  - Unique and primary
  - Functional dependency
- Indexes
- Triggers
- Sequences
- Functions and procedures except for external routine executables
- Object authorizations, privileges, security, access control, and audit configuration
- Table statistics, profiles, and hints
- Packages

**The following components of a schema are not created on the target database:**

- Aliases
- Created global variables
- Jobs
- Index extensions
- Hierarchical tables, typed tables, and typed views
- Nicknames
- Structured types
- Methods
- Servers
- Wrappers
- Functional mappings and templates
- OLE DB external functions
- Sourced procedures
- External routine executable files
- System catalogs
- Range partitioned tables

# Transportable Schemas: Example



The database contains the following valid transportable sets:

- **mydata1:** schema1 + schema2
- **mydata2 + myindex:** schema3
- **multidata1 + multiuser2 + multiindex1:** schema4 + schema5
- Any combination of the above transportable sets

**To Move All Table Spaces:**

```
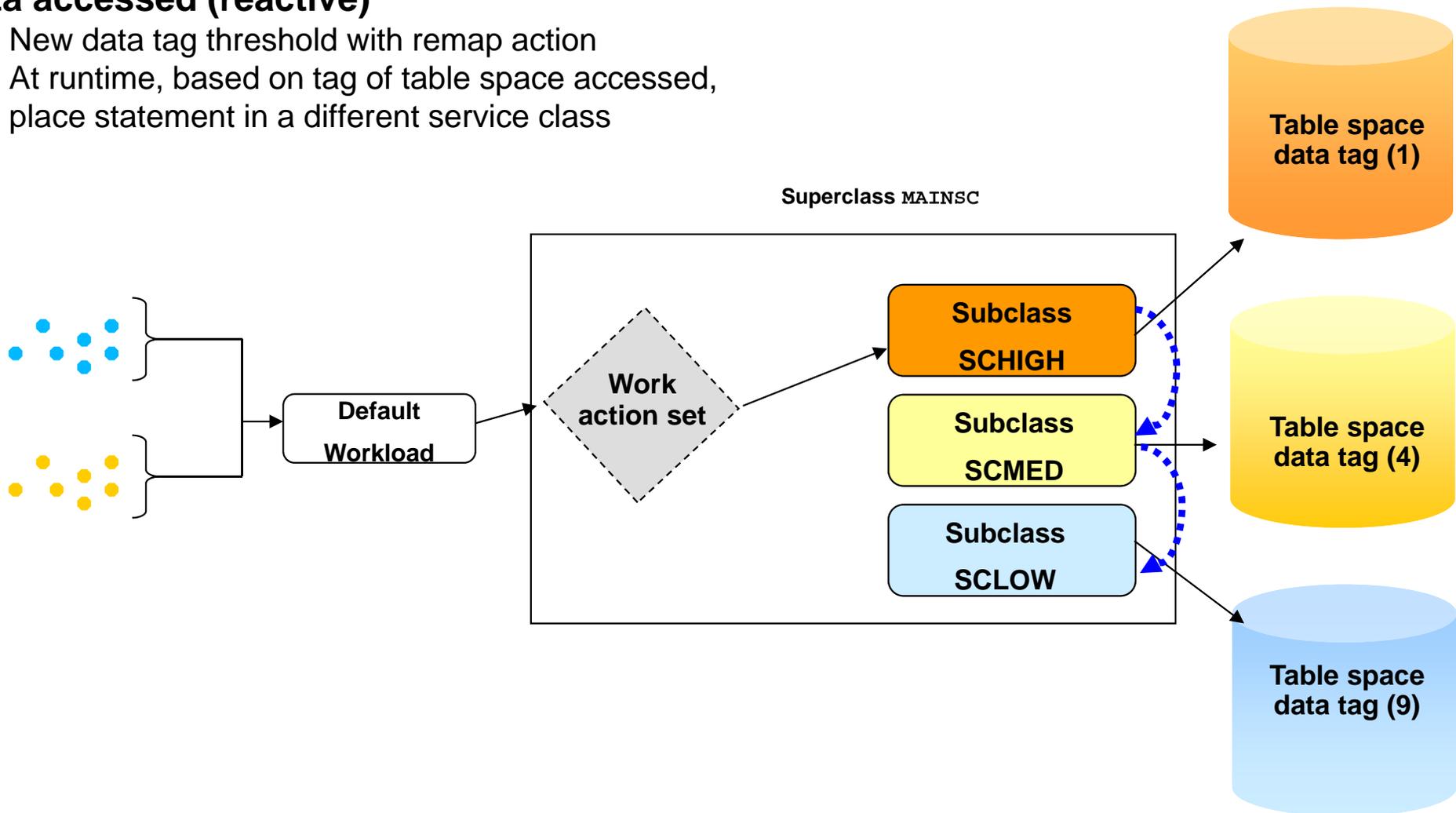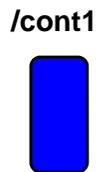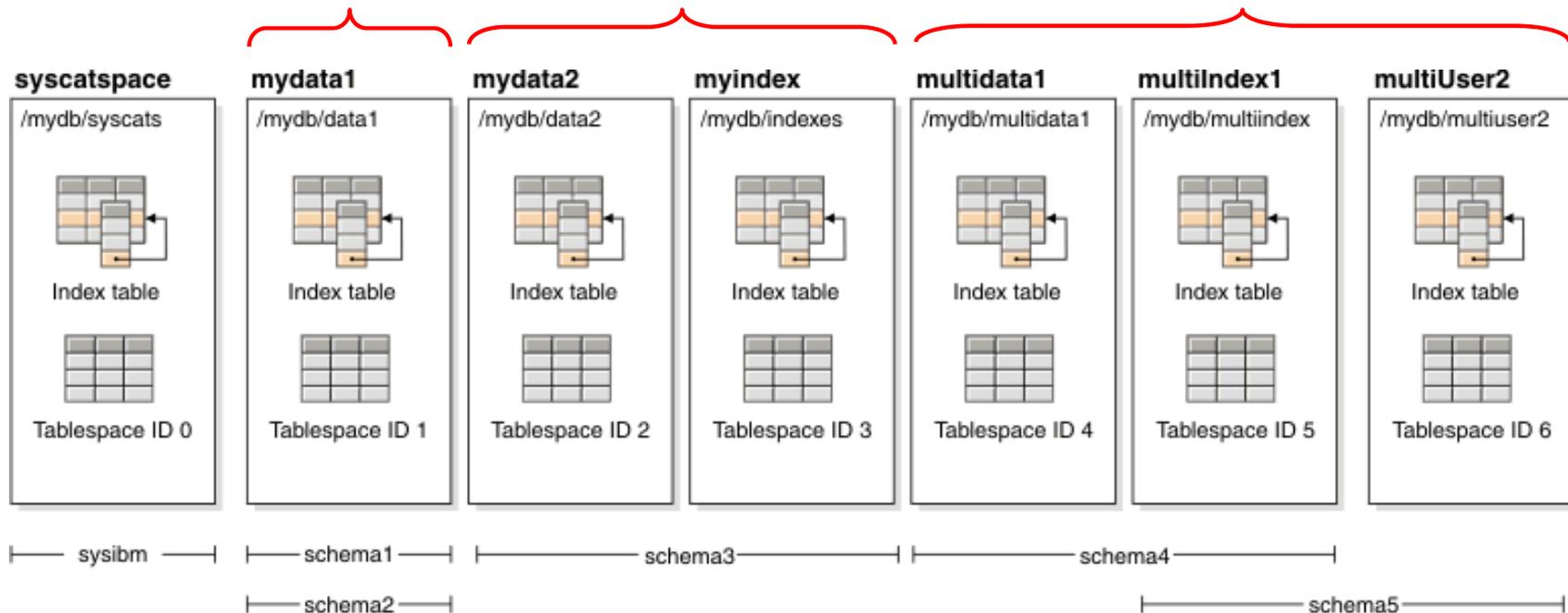restore db old_db \
  tablespace ("mydata1","mydata2","myindex","multidata1","multiindex1","multiuser2")\
  schema ("schema1","schema2","schema3","schema4","schema5") transport into new_db
```

Agenda

- **Overview of Automatic Storage and Feature Highlights**

- **Multi-Temperature Data and Storage Management**

- **Moving To Automatic Storage**

- **Best Practices**

Best Practices

- **Use Automatic Storage for everything-everywhere striping**
  - **DB2 10.1:** Use storage groups for finer grained control and for improved multi-temperature storage and data management

- **For each storage group**
  - Create LUN(s) based on physical disk requirements (MB/s, IOPS)
  - Create a file system per LUN, specifying each as a storage path
  - When defining/adding storage paths, ensure that the file systems associated with the paths are approximately equal in capacity and have the same media characteristics
  - Set the overhead and device read rate attributes to reflect the capabilities of the underlying storage

- **Allow a table space to inherit media attributes and data tag from its storage group**
  - Won't need to set them if a table space is moved to a different storage group
  - Storage group's DEVICE READ RATE gets converted to TRANSFERRATE for the table space

- **See the DB2 Database Storage Best Practices document:**
  - http://www.ibm.com/developerworks/data/bestpractices/databasestorage/

Best Practices (cont.)

- **If implementing a multi-temperature storage strategy for your data:**

  - Use range partitioning

  - Assign partitions to table spaces such that the granularity of the data stored in each table space has the same temperature change schedule
    - Remember that all of the data in the table space is moved when a table space is moved from one storage group to another one

  - Consider backing up the table spaces with a frequency based on the temperature of the data

  - Do not change the storage group for too many table spaces at once
    - A rebalance is started for each table space that is moved
      - Heavy I/O impact (but rebalance can be throttled or suspended if necessary)
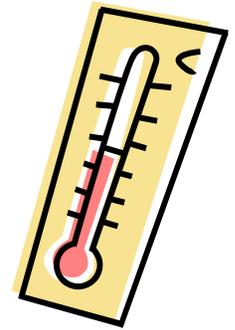
For More Information

- **White Paper: DB2 10.1 Multi-Temperature Data Management Recommendations**
  - http://www.ibm.com/developerworks/data/library/long/dm-1205multitemp/index.html

- **DB2 Information Center: Data Management Using Multi-Temperature Storage**
  - http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/topic/com.ibm.db2.luw.admin.dbobj.doc/doc/c0059106.html

- **DB2 Best Practices: Upgrading to the DB2 pureScale Feature** *(details on moving to AS)*
  - http://www.ibm.com/developerworks/data/bestpractices/purescaleupgrade

- **DB2 Best Practices: Database Storage**
  - http://www.ibm.com/developerworks/data/bestpractices/databasestorage/

Thank You