IBM® DB2® for Linux®, UNIX®, and Windows®

DB2 V10.1 Multi-temperature Data Management Recommendations

Jim Seeger
Senior Software Engineer, DB2 for Linux, UNIX, and Windows development

Naresh Chainani
DB2 for Linux, UNIX, and Windows Software Developer

Aruna De Silva
Quality Assurance & Solutions Specialist, DB2 Distributed and Data Warehousing

Karen Mcculloch
Manager, DB2 WLM Development

Kiran Chinta
DB2 for Linux, UNIX, and Windows Software Developer

Vincent Kulandai Samy
DB2 for Linux, UNIX, and Windows Software Developer

Tom Hart
DB2 for Linux, UNIX, and Windows Software Developer

Issued: April 2012

Executive summary 4

Introduction 5

Key concepts 8

Determining how many storage groups to define 9

Setting storage media attributes 10

  Determining media attributes 10

  Impact on query optimizer planning 10

Database design for multi-temperature data 11

  Creating storage groups 12

  Creating table spaces 14

  Retrieving table space details 15

Moving data between temperature tiers 16

  Command line example 16

  Use InfoSphere Optim Configuration Manager for data migration 17

Managing the impact of data movement 20

Upgrading an existing database into a multiple temperature storage database 22

Using data tags and DB2 workload manager to prioritize activities based on the data that is being accessed 24

  Using data tags with DB2 WLM predictively 25

  Using Data Tags with DB2 WLM reactively 29

Backup and recovery considerations 33

Setting up an alternative copy of your database 33

Appendix A: Determining media attributes by using the static approach 36

  Determining the file systems used as storage paths in a database 36

  Computing DEVICE READ RATE or TRANSFERRATE and OVERHEAD attribute values 37

Appendix B: Determining a device serial number on Linux systems 39

  Mapping the storage paths to LUNs (host addressable disk device) ... 39

Finding the serial numbers of the hdisks (LUNs) 39

Appendix C: Determining a device serial number on AIX systems........... 41

Mapping the storage paths to LUNs (host addressable disk device)... 41

Finding the serial numbers of the hdisks (LUNs) 41

Appendix D: Determining media attributes by using the dynamic approach.. 42

Using I/O performance monitoring tools to gauge disk throughput... 42

Conclusion 44

Further reading.. 45

Contributors.. 45

Notices 47

Trademarks 48

Executive summary

Data in a data warehouse can be classified according to its temperature. The temperature of data is based on how old it is, how often it is accessed, how volatile it is, and how important the performance of the queries that access the data is. Hot data is frequently accessed and updated, and users expect optimal performance when they access this data. Cold data is rarely accessed and updated, and the performance of the queries that access this data is not essential. 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.

This paper presents a strategy for managing a multi-temperature data warehouse by storing data on different types of storage devices based on the temperature of the data. It provides guidelines and recommendations for each of the following tasks:

- Identifying and characterizing data into temperature tiers
- Designing the database to accommodate multiple data temperatures
- Moving data from one temperature tier to another
- Using DB2® workload manager to allocate more resources to requests for hot data than to requests for cold data
- Planning a backup and recovery strategy when a data warehouse includes multiple data temperature tiers

The content of this paper applies to data warehouses based on version 10.1 or later of DB2 Database for Linux, UNIX, and Windows.

Introduction

The quantity of data stored in data warehouse environments is growing at an unprecedented rate. There are several reasons for this growth. For example:

- Database users are retaining enormous amounts of detailed data such as transaction history, web search queries, and detailed phone records.
- As data mining algorithms continue to improve, and as increasing processing power becomes available, organizations are analyzing much older historical data to predict future trends more accurately.
- Stricter regulations and audit standards now require businesses to keep data for longer periods of time than previously.
- Many businesses are eliminating the cost of keeping paper-based records by switching to web-based records.

However, not all of the data in a data warehouse is equally valuable to an organization. In general, the most recent data in a warehouse is much more likely than older data to be accessed by queries and maintenance processes or to be updated. Such data is therefore called *hot*. As time goes by, data tends to *cool off*, becoming *warm* and later *cold*, meaning that the probability that users access or update this data significantly decreases. The data must still be available, however, for regulatory requests, audits, and long-term research. Another important characteristic of requests for colder data is that users do not typically insist on optimal performance for these requests. Because strong performance for these queries is not essential, you can place colder data on slower, less expensive types of storage devices.

A warehouse can contain several different temperature tiers (hot, warm, cold, dormant). In general, the number of temperature tiers is tied to the number of different types of storage devices that are attached to the warehouse. For example, you might store hot data on new, solid-state drives (SSD); warm data on new, fast magnetic storage devices; and cold and dormant data on older, less efficient magnetic storage devices.

The definition of each data temperature depends on the specific environment, but data temperatures usually fall into fairly common categories. The following chart provides some guidelines for classifying data by temperature:

Data temperature	Data temperature characteristics	Typical data age
Hot	Tactical and OLTP type data – current data that is accessed frequently by queries that must have short response times. For example, high volume, small result set point queries in operational data stores (ODS).	0 - 3 months and aggregates or summaries of this data
Warm	Traditional decision support type data – data that is accessed less frequently and by queries that most likely do not require short response times.	3 - 13 months and aggregates or summaries of this data
Cold	Deep historical and legacy data – data that is typically accessed infrequently.	13 months - 5 years
Dormant	Regulatory type or archival data – data that is accessed infrequently and that is never updated.	Over 5 years

As data ages, the ***average*** temperature of the data tends to cool off. There can be temperature fluctuations, or hot spots, as users perform periodic analysis, such as an analysis of the current quarter compared to the same quarter last year. But typically a small proportion of the data in a warehouse is considered hot or warm and 70% to 90% of the data is considered cold or dormant. The following diagram shows a typical distribution of data across temperature tiers.

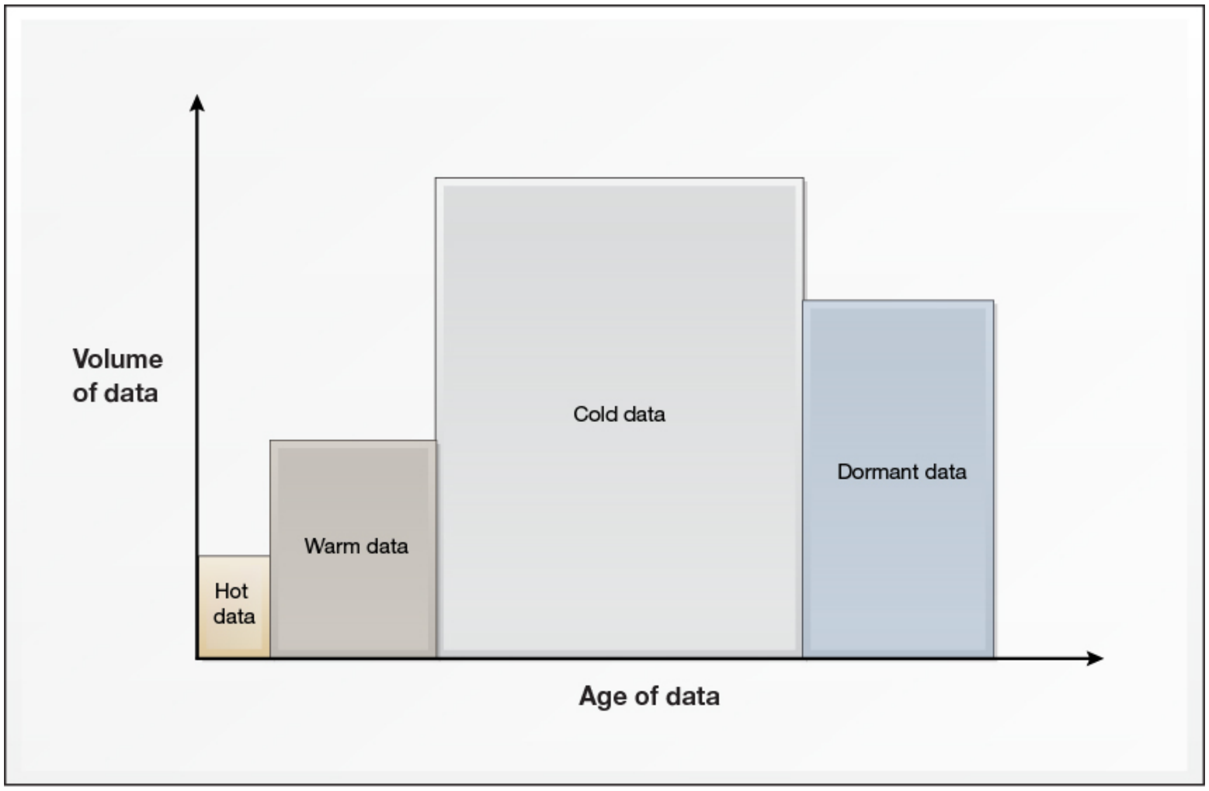


Figure 1: Typical distribution of data across temperature tiers

Classification of data into temperature tiers is also affected by the business rules that dictate how your organization moves data between the parts of the data warehouse that support the data of a particular temperature. For example, your organization's reports focus on quarterly data and always compare the net sales at the end of each fiscal month to the net sales during the same month for the previous year. Such a business rule implies that the definition of hot data should include the most recent three fiscal months and warm data should include the next 10 fiscal months. When the end of the month reporting is performed, the 13th fiscal month can be moved back into hot storage so your reports run optimally. After the reports are complete, the 13th fiscal month can be moved back to warm or cold storage.

DB2 V10.1 introduces storage groups wherein each storage group represents a tier of storage in the warehouse. A storage group provides for logical grouping of paths on storage subsystems with similar attributes. By associating automatic storage table spaces with a storage group, database administrators can place data that is accessed more frequently into faster storage while data that is accessed less often is still available, albeit with a slower response time. A typical warehouse has 2 - 5 tiers of storage based on the range of storage devices the organization has.

As time goes by, newer data is ingested in the warehouse. The existing hot data is no longer accessed as much as before and needs to make way for the newly ingested data. This is achieved by the DBA changing the storage group that the table space is currently associated with. While the table space data moves from hot storage to warm storage, the data continues to be available to queries. The actual data movement happens asynchronously and can be suspended to allow for more important query workloads to run faster. The asynchronous data movement can then be resumed later when the reporting workload on the system is light.

The DB2 optimizer factors in the I/O properties of the various storage tiers into the costing model. This necessitates that the DBA set the media properties like DEVICE READ RATE and OVERHEAD for a storage group to reflect the capability of the storage subsystems. The table spaces using the storage group then automatically inherit the media attributes from the corresponding storage group. Provided that the media attribute settings reflect the order of magnitude differences between the storage subsystem capabilities, the exact value for these attributes are generally not that important.

By using redirected restore, the DBA can change storage paths for a storage group. This functionality can be useful when cloning the production database to a test system where the storage paths can be redirected to reflect the storage subsystem topology on the test system.

This paper provides details on how organizations can adopt and leverage the DB2 multi-temperature capabilities and integrate policies by using DB2 workload management (WLM) to have a high performing warehouse where important queries that touch hot data get the most resources and complete faster, thus assisting businesses with timely decision making.

Key concepts

This paper refers to the following key concepts:

A *database partition* is a portion of a database that consists of its own data, indexes, configuration files, and transaction logs. A partitioned database environment is a database installation that supports the distribution of data across database partitions.

A *database partition group* is a set of one or more database partitions in a database that should have a common distribution map.

A *storage group* is a named set of storage paths on which DB2 for Linux, UNIX, and Windows data can be stored. The storage paths have a common set of media attributes.

A table space is a storage structure that can contain tables, indexes, large objects, and long data. Table spaces organize data in a database into logical storage groupings that relate to where data is stored on a system. A table space can belong to only a single database partition group. Automatic storage managed table spaces can be assigned to a specific storage group based on the temperature of the data in the table space.

Table partitioning is a data organization schema in which the data in a table is partitioned across multiple storage objects. Each unit of a partitioned table is called a *data partition* (or sometimes *range*). In most cases, the data partition is defined based on a time dimension, such as month or quarter. A table that is partitioned into data partitions is called a *partitioned table*.[1]
The main benefit of using table partitioning is the ability to attach ("roll in") and detach ("roll out") data partitions almost instantaneously. Beginning with DB2 V9.7 databases, the table partitioning feature also provides the ability to partition indexes.

[1] Do not confuse table partitioning with database partitioning. They are separate data organization schemas. You can use one, both, or neither of these schemas. With table partitioning, a table is split into *data partitions*; with database partitioning, a database is split into *database partitions*. One or more data partitions can be stored on a single database partition. In this paper, the term *partitioned table* always refers to a table that is split using table partitioning.

Determining how many storage groups to define

There are several factors to consider when you determine how many storage groups to define for your DB2 instance. You might have multiple storage tiers available that provide different levels of service. That service can be related to performance or capabilities of the storage device. In figure 2, we have three tiers of storage (SSD, Fibre Channel System Attached Storage and SATA) in which we want to store our sales table. Therefore we defined three storage groups.

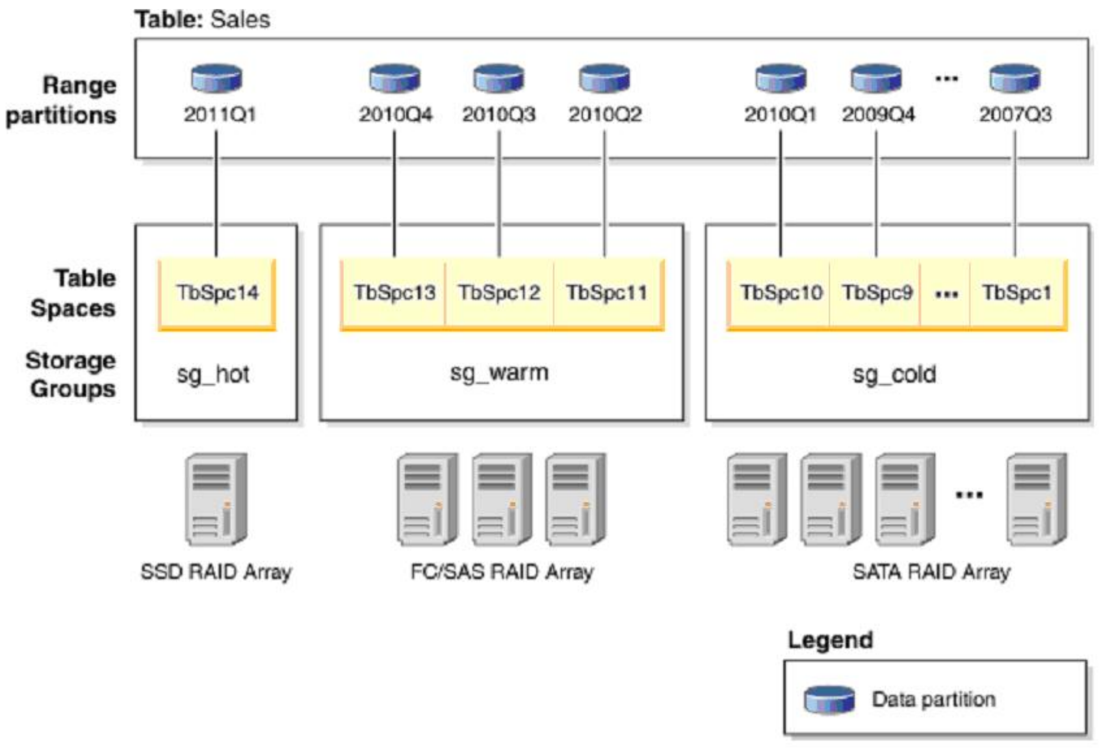


Figure 2: Multiple temperature topology example

The best response times are needed when accessing the most recent quarter's data. Thus, we place it on the fastest storage type. The response times of the next three most recent quarters needs to be good, but not as fast as the current quarter. Finally, the remainder of the data is not accessed as frequently and the response time can be longer. In such a configuration, most of the data is stored on our least expensive storage while allowing our service level agreements on the more recent data to be satisfied. Therefore, the overall Total Cost of Ownership for the warehouse is improved.

Setting storage media attributes

There is a wide range of media types available in the market today, including hard disk drives, memory cards, and solid-state drives. Some have good read performance, some have good write performance, and some have good read/write performance compared to others. When it comes to organizing your data, it is critical to think about where the data is placed and when, and for how long the data is stored on the selected media devices. Media types vary in performance and the highest performing media types come at a cost. Taking the best total cost of ownership approach, data can be distributed over various media devices.

Determining media attributes

In general, there are two approaches to customizing DEVICE READ RATE and OVERHEAD storage group attributes (or TRANSFERRATE and OVERHEAD table space attributes), when the storage paths are created on file systems backed by external storage devices.

- Static (using the published specs): trace the LUN (seen by the host) back to the external storage controller and determine the actual back-end device or devices the LUN is created from. When you have that information, search for the model number in the disk drive manufacturer's website to find the published drive spec sheet, which can then be used to compute these media attributes.

- Dynamic (using I/O profiling and estimation techniques): use a tool such as I/O meter (or a similar tool supplied by the storage vendor) to measure I/O throughput, and use the observations to estimate. However, you still need to know the back-end physical disk specs to set the OVERHEAD value.

A host addressable external disk device (or LUN) can be either a directly attached storage controller or can be a storage area network (SAN). For example, host systems can be attached directly to controller A and controller B of a DS5300 Storage system by using two cables connected to two Fibre Channel (FC) initiators at the back of each host system. Similarly, the host systems can be accessing disks from an EMC V-Max Storage controller by using a SAN fabric, where both hosts systems and the storage controller is connected to the same or interconnected SAN fabric switch or switches and mapped to the same zone. A zone controls which initiators can log in to or access which end-ports.

For more information, see Appendix A which describes the static approach and Appendix D which describes the dynamic approach.

Impact on query optimizer planning

When a query is compiled, the DB2 optimizer calculates several plans and selects the best plan by considering various statistics of the table data related to the media devices like OVERHEAD and TRANSFERRATE. In previous releases, the storage was likely uniform in its capabilities and you did not have to worry about setting the media attributes. Now with the support of multiple tiers of storage, setting the media attributes is more relevant.

If you implement cost-based workload management and you use static SQL packages, rebind the packages. Rebind the packages because the cost of the queries is pre-calculated and stored in the package.

For best query performance, set the storage group media device attributes OVERHEAD and DEVICE_READ_RATE to reflect the storage path's capabilities.

Database design for multi-temperature data

When you design a database for multi-temperature data, the main principle recommended in this paper is to physically separate hot, warm, cold, and dormant data and to isolate the different temperature tiers in different storage groups. Place hot and warm data in table spaces that reside on fast storage and cold and dormant data in table spaces that reside on less expensive, slower storage devices. This type of database design makes all data accessible but optimizes the price-to-performance balance by using lower-cost storage for the data that is rarely accessed or updated.

Physically separate hot, warm, cold, and dormant data by storing data in table spaces based on the temperature of the data. Store your hot data on fastest storage, warm data on fast storage and cold and dormant data on slower storage.

Refer to the following recommendations when you are designing tables for multi-temperature data:

- Use table partitioning and organize your data partitions into table spaces based on how granularity you plan to move data from one temperature tier to another. For example, if you plan to move your data from hot to cold storage on a quarterly basis, your table space granularity should be quarterly.

  Depending on your scenario, you might choose one of the following methods to manage data in your table spaces:
    - Place each data partition into a separate table space.
    - Place multiple data partitions into a single table space.

  The advantage of the first method is a simpler data model; the advantage of the second method is to limit the number of table spaces in your environment, which can reduce administrative work.

- Define the table spaces for each temperature tier to be in the same database partition group. The reason for keeping all the table spaces in the same database partition group is to support collocated joins, which provide better query performance than non-collocated joins.

- When you name table spaces or data partitions for multi-temperature tables, use naming conventions that indicate the date range of the data in the particular table space, table space container, or data partition.

Use table partitioning and arrange table partitions in table spaces so that the granularity of the data stored in each table space matches your schedule for moving data from one temperature tier to another.

Creating storage groups

A storage group is a named set of storage paths where data can be stored. Storage groups are configured to represent different classes of storage available to your database system. You can assign table spaces to the storage group that best suits the data access pattern. Only automatic storage managed table spaces can use storage groups.

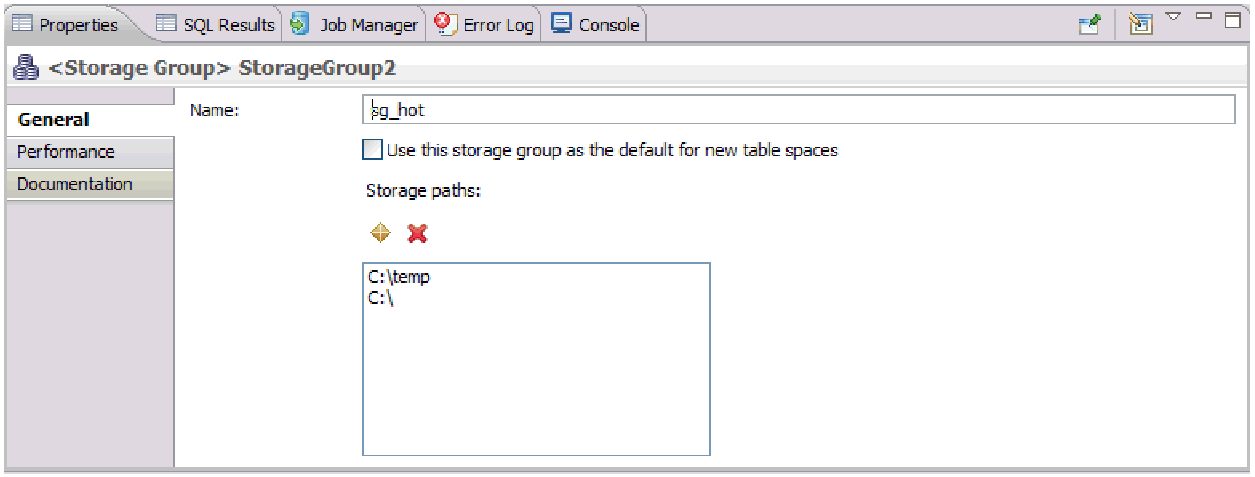
To create a storage group by using the DB2 command line, enter the following statement:

```
CREATE  STOGROUP  sg_warm  ON  '/db2/warm01',  '/db2/warm02',
 '/db2/warm03' ... DEVICE READ RATE 100 OVERHEAD 6.725
```

Where sg_*warm* is the name of the storage group and */db2/warm01, /db2/warm02, /db2/warm03, …* are the storage paths to be added.

To help ensure predictable performance, all the paths that you assign to a storage group should have the same media characteristics: latency, device read rate, and size.

The steps to perform the same action in Data Studio V3.1 follow in figures 3 and 4.



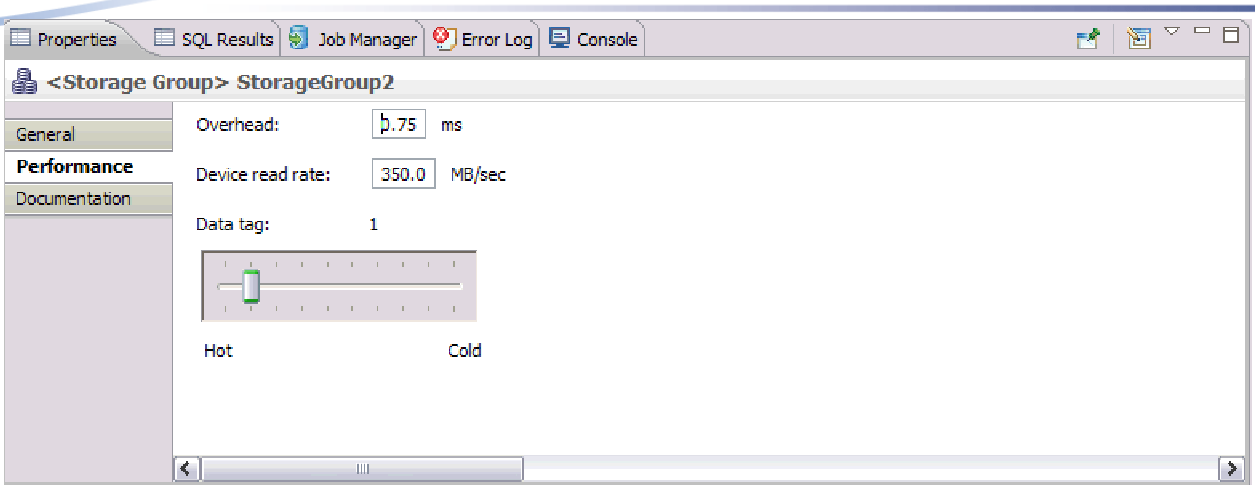
Name: sg_hot

Use this storage group as the default for new table spaces

Storage paths:

C:\temp
C:\

Figure 3: Data Studio panel to create the storage group and its storage paths.



<Storage Group> StorageGroup2

Overhead: 0.75 ms

Device read rate: 350.0 MB/sec

Data tag: 1

Figure 4: Data Studio panel to define the storage group attributes.

Retrieving storage group details

The administrator can obtain storage group details by querying from the SYSCAT.STOGROUPS view and table function ADMIN_GET_STORAGE_PATHS as follows:

```
SELECT varchar(sgname, 12) AS name, sgid,  DEFAULTSG,
   cast(DEVICEREADRATE as dec(5,1)) READRATE,
   cast(OVERHEAD as dec(5,3)) OVERHEAD, DATATAG,
   varchar(DB_STORAGE_PATH, 40) AS STORAGE_PATH
FROM SYSCAT.STOGROUPS, table (ADMIN_GET_STORAGE_PATHS(NULL, -1))
WHERE sgid = storage_group_id
```

```
NAME         SGID  DEFAULTSG READRATE OVERHEAD DATATAG STORAGE_PATH

------------ ----- --------- -------- -------- ------- ------------

SG_HOT           1         N      350     0.75       1 /db2/hot01

SG_WARM          2         Y      100    6.725       4 /db2/warm01

SG_WARM          2         Y      100    6.725       4 /db2/warm02

SG_WARM          2         Y      100    6.725       4 /db2/warm03

SG_COLD          3         N       67      7.5       7 /db2/cold01

SG_COLD          3         N       67      7.5       7 /db2/cold02

SG_COLD          3         N       67      7.5       7 /db2/cold03

SG_COLD          3         N       67      7.5       7 /db2/cold04

SG_COLD          3         N       67      7.5       7 /db2/cold05
```

Creating table spaces

The following steps provide more details on how to set up multi-temperature data storage for the sales data in the current fiscal year:

Create two storage groups to reflect the two classes of storage, a storage group to store hot data and a storage group to store warm data.

```
CREATE STOGROUP sg_hot ON '/db2/hot01’ DEVICE READ RATE 350
   OVERHEAD 0.75
CREATE STOGROUP sg_warm ON '/db2/warm01’, '/db2/warm02’,
'/db2/warm03’
```

Create four table spaces, one per quarter of data in a fiscal year, and assign the table spaces to the storage groups.

```
CREATE TABLESPACE tbsp_2011q2 USING STOGROUP sg_warm
CREATE TABLESPACE tbsp_2011q3 USING STOGROUP sg_warm
CREATE TABLESPACE tbsp_2011q4 USING STOGROUP sg_warm
CREATE TABLESPACE tbsp_2012q1 USING STOGROUP sg_hot
```

This association results in table spaces that inherit the storage group media properties.

When an automatic storage table space inherits the TRANSFERRATE setting from the storage group it is using, the DEVICE READ RATE of the storage group, which is in megabytes per second, is converted into milliseconds per page read accounting for the PAGESIZE setting of the table space.

The conversion formula follows:

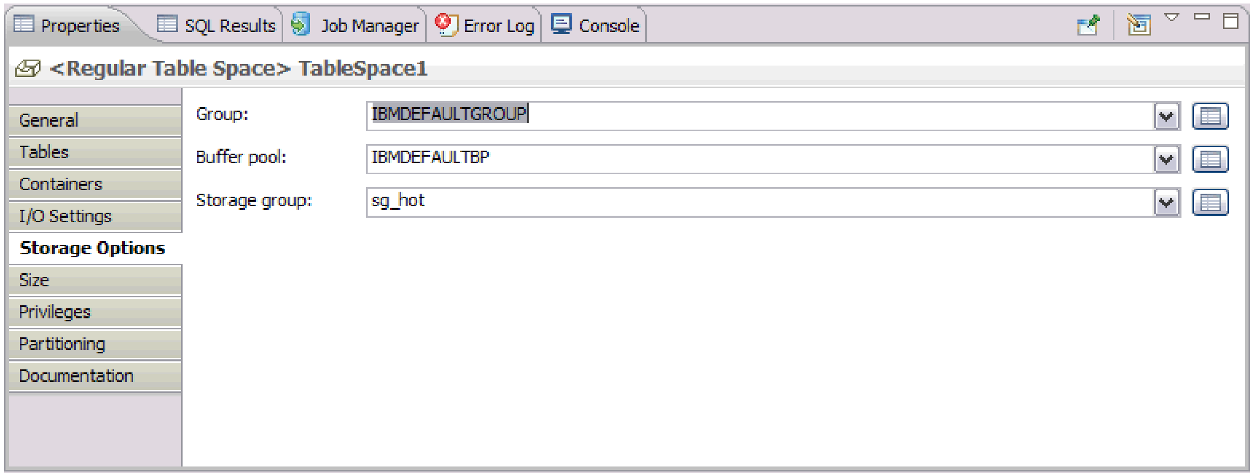
TRANSFERRATE = (1 / DEVICE READ RATE) * 1000 / 1024000 * PAGESIZE

Set up your range partitions in your sales table.

```
    CREATE TABLE sales (order_date DATE, order_id INT, cust_id
BIGINT)
    PARTITION BY RANGE (order_date)
    (PART "2011Q2" STARTING ('2011-04-01') ENDING ('2011-06-30')
in "tbsp_2011q2",
     PART "2011Q3" STARTING ('2011-07-01') ENDING ('2011-09-30')
in "tbsp_2011q3",
     PART "2011Q4" STARTING ('2011-10-01') ENDING ('2011-12-31')
in "tbsp_2011q4",
     PART "2012Q1" STARTING ('2012-01-01') ENDING ('2012-03-31')
in "tbsp_2012q1");
```

The 2011Q1 data represents the current fiscal quarter and is using the *sg_hot* storage group.

In Data Studio V3.1, you can specify the storage group as part of the table space properties at creation time as can be seen in figure 5.



<Regular Table Space> TableSpace1

Group: IBMDEFAULTGROUP

Buffer pool: IBMDEFAULTBP

Storage group: sg_hot

Figure 5: Example of creating a table space in Data Studio

Retrieving table space details

DB2 V10.1 introduces the ability for automatic storage table spaces to inherit their transfer rate, overhead, and data tag settings from the storage group they are associated with. This means a -1 value, which indicates inherit, can be stored in the SYSCAT.TABLESPACES for these attributes. For example, the following query shows all the table spaces in the system and their attribute's effective values.

```
SELECT varchar(tbspace, 20) AS name, tbspaceid, sgid,
   cast(CASE WHEN a.datatag = -1 THEN b.datatag ELSE a.datatag END
   AS smallint) datatag,
   cast(CASE WHEN a.overhead = -1 THEN b.overhead ELSE a.overhead
   END AS decimal(5,3)) overhead,
   cast(CASE WHEN a.transferrate = -1 THEN (1 / b.devicereadrate)
   / 1024 * a.pagesize else a.transferrate END AS decimal(5,3))
   transferrate
FROM syscat.tablespaces a LEFT OUTER JOIN syscat.stogroups b ON
a.sgid = b.sgid
```

```
NAME                 TBSPACEID  SGID DATATAG OVERHEAD TRANSFERRATE
-------------------- ---------  ---- ------- -------- ------------
SYSCATSPACE                  0     0       0    6.725        0.040
TEMPSPACE1                   1     0       0    6.725        0.040
```

```
USERSPACE1                    2      0      0     6.725         0.040

SYSTOOLSPACE                  3      0      0     6.725         0.040

TBSP_2012Q1                   4      3      1     0.750         0.011

TBSP_2011Q4                   5      2      4     6.725         0.040

TBSP_2011Q3                   6      2      4     6.725         0.040

TBSP_2011Q2                   7      1      7     7.500         0.059


  8 record(s) selected.
```

Moving data between temperature tiers

As the data ages and cools off over time, it is important to move it from one temperature tier to another. For example, you can set up a batch job that moves data from one temperature tier to another every month or quarter. Because data in different temperature tiers is stored on separate storage devices, moving the data results in copying the data from the source storage group's storage paths to the target storage group.

If you have multiple tables that share a table space, ensure that the tables have the same temperature characteristics. All the data in a table space is moved when its storage group is altered.

After you move the data from one temperature tier to another, collect statistics by using the RUNSTATS command.

The following procedure shows how to move the current data partition from a storage group that hosts hot data to a storage group that hosts warm data. This example is based on the assumption that, every quarter, you move your data from hot storage to warm storage, and that you organized your data such that the data for each quarter is stored in a separate table space.

Command line example

Referring to our earlier example in the section titled "Database design for multi-temperature data", let's see what the steps are when a new fiscal quarter begins. After the current quarter passes, create a table space for the new quarter, and assign the table space to the *sg_hot* storage group.

```
  CREATE TABLESPACE tbsp_2012q2 USING STOGROUP sg_hot
```

Move the table space for the quarter that just passed to the *sg_warm* storage group. To change the storage group association for the tbsp_2012q1 table space, issue the ALTER TABLESPACE statement with the USING STOGROUP option.

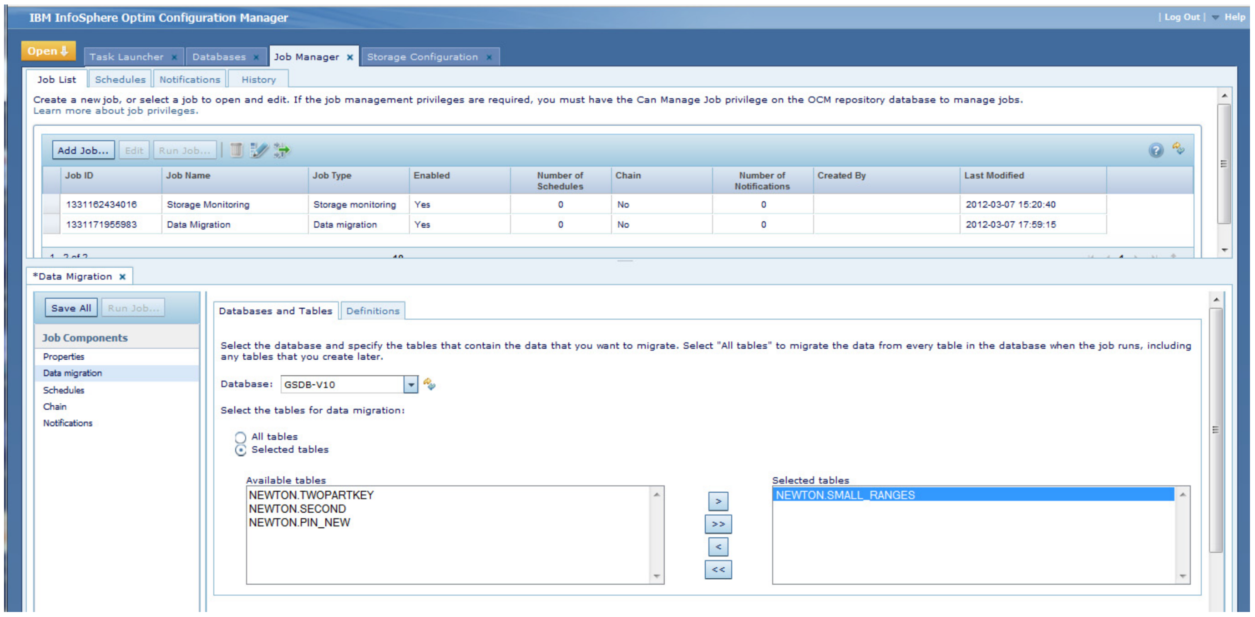
```
ALTER TABLESPACE tbsp_2012q1 USING STOGROUP sg_warm
```

At this point, the data stored in *tbsp_2012q1* begins to move to storage group *sg_warm* in the back ground. The data remains available to applications during this process. Since the *tbsp_2012q1* table space inherits its media attributes from its storage group, its media attributes now are based on storage group *sg_warm*.

Do not overload your system by moving too many table spaces at one time. Each ALTER TABLESPACE … USING STOGROUP … statement initiates an implicit background REBALANCE operation at COMMIT time to move the table space data to the specified storage group.

Use InfoSphere Optim Configuration Manager for data migration

InfoSphere Optim Configuration Manager includes a job manager that you can use to define one or more data migration jobs. The following example shows a job that is being run immediately, but you can also define recurring jobs to run on specific dates and times. When you create a data migration job and after you select the database you want to work with, you are presented with a list of partitioned tables where the first range partitioning column is of type date and all table spaces used by the table are of type automatic storage managed. Now select the tables you are interested in adding to the job, as shown in figure 6.



Create a new job, or select a job to open and edit. If the job management privileges are required, you must have the Can Manage Job privilege on the OCM repository database to manage jobs.
Learn more about job privileges.

Job ID	Job Name	Job Type	Enabled	Number of Schedules	Chain	Number of Notifications	Created By	Last Modified
1331162434016	Storage Monitoring	Storage monitoring	Yes	0	No	0		2012-03-07 15:20:40
1331171955983	Data Migration	Data migration	Yes	0	No	0		2012-03-07 17:59:15

Select the database and specify the tables that contain the data that you want to migrate. Select "All tables" to migrate the data from every table in the database when the job runs, including any tables that you create later.

Database: GSDB-V10

Select the tables for data migration:

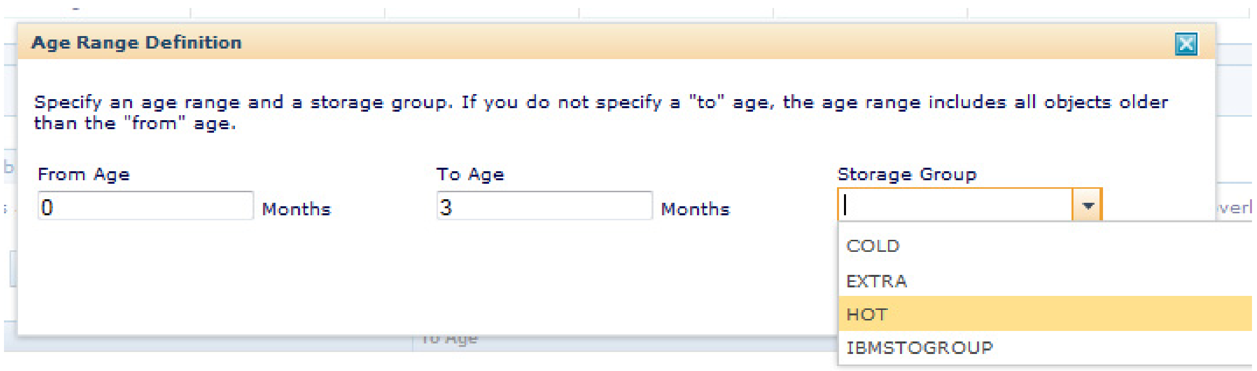
( ) All tables
[✓ Selected tables]

Available tables
NEWTON.TWOPARTKEY
NEWTON.SECOND
NEWTON.PIN_NEW

Selected tables
NEWTON.SMALL_RANGES

Figure 6: Defining a data migration job

After you select your tables, you can specify the data migration definition. You now specify which storage group you want specific age ranges of your data to reside. In the example shown in figure 7, the data that is 0 - 3 months old resides in the *hot* storage group. You cannot define overlapping age ranges in your aging policies.

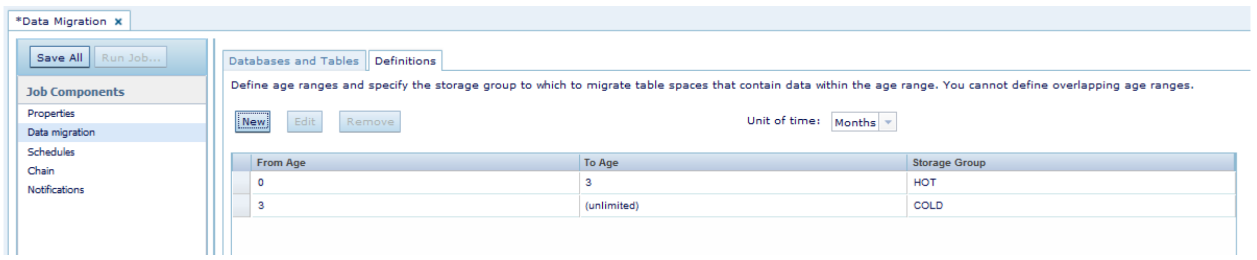

Age Range Definition

Specify an age range and a storage group. If you do not specify a "to" age, the age range includes all objects older than the "from" age.

From Age	To Age	Storage Group
0 Months	3 Months	

Figure 7: Defining a data migration aging policy.

The Data Migration panel shows the defined age ranges for the job that is being created. You can also edit or remove existing age range definitions. Another thing to highlight is the ability to select the unit of time used for the age range definitions. The details in figure 8 show two age range definitions where data that is 3 months or less is uses HOT storage and all other data is placed in COLD storage.



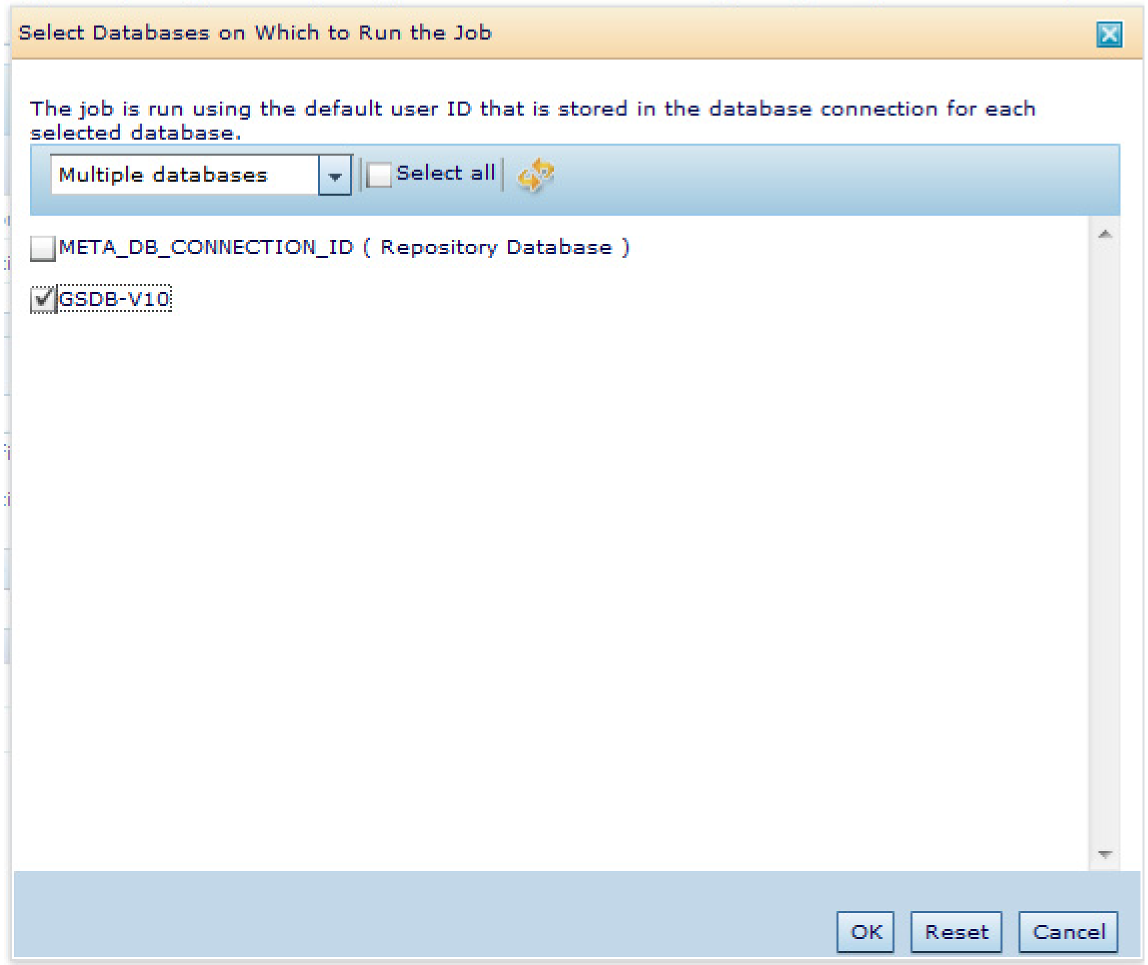
Define age ranges and specify the storage group to which to migrate table spaces that contain data within the age range. You cannot define overlapping age ranges.

Unit of time: Months

From Age	To Age	Storage Group
0	3	HOT
3	(unlimited)	COLD

Figure 8: Migration Definition age range definitions.

The next step is to define which DB2 databases we want this job to run, as seen in figure 9.


Select Databases on Which to Run the Job

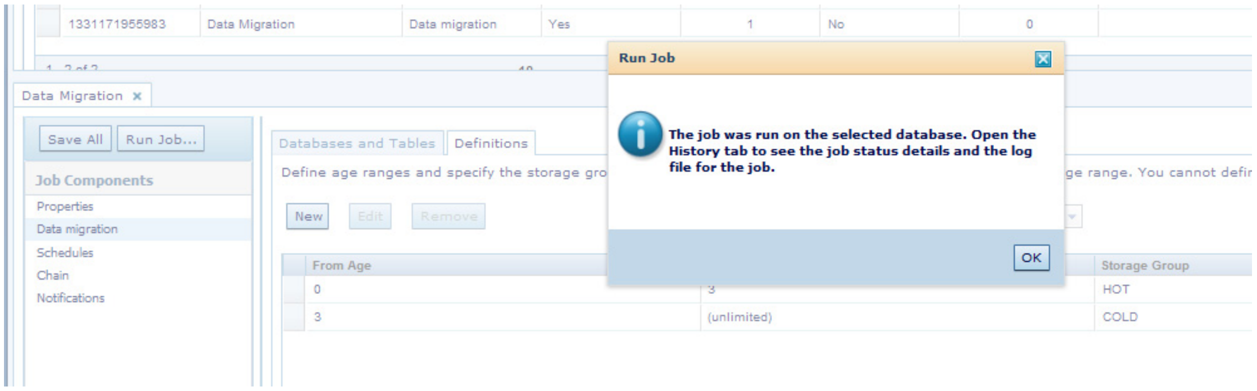
The job is run using the default user ID that is stored in the database connection for each selected database.

[ ] META_DB_CONNECTION_ID (Repository Database)

[✓] GSDB-V10

Figure 9: Choosing which database to run your job.

Figure 10 shows an alert message when the data migration job finishes.


1331171955983	Data Migration	Data migration	Yes	1	No	0

Define age ranges and specify the storage gro... ...ge range. You cannot defi...

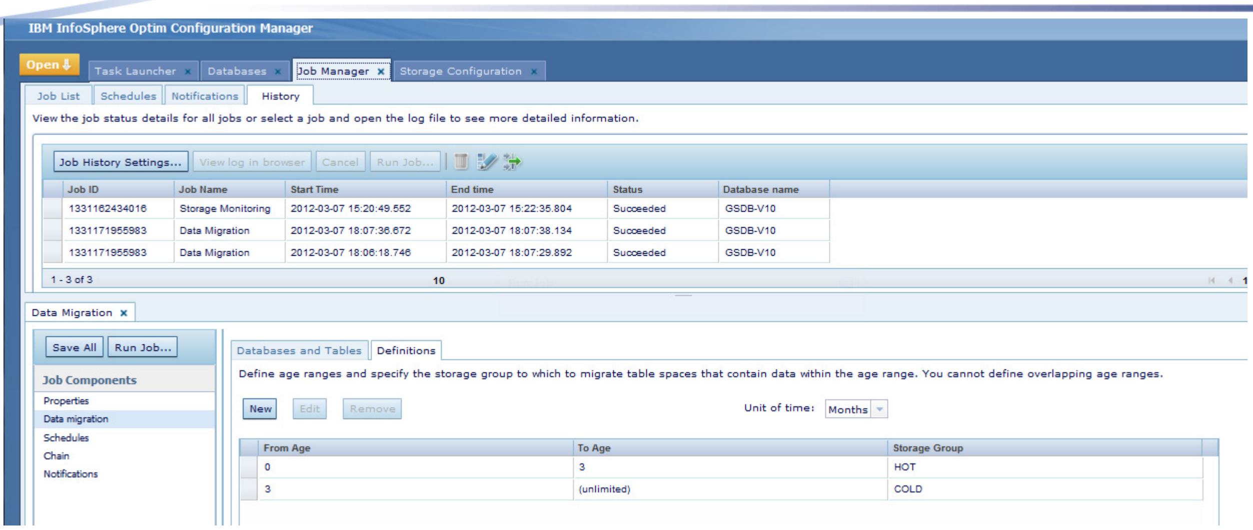
From Age		Storage Group
0	3	HOT
3	(unlimited)	COLD

Run Job

The job was run on the selected database. Open the History tab to see the job status details and the log file for the job.

Figure 10: Running the data migration job

A job history file is created when a job runs. Open the job log file to see what actions the job performed. Figure 11 shows the job history listing.


IBM InfoSphere Optim Configuration Manager

View the job status details for all jobs or select a job and open the log file to see more detailed information.

Job ID	Job Name	Start Time	End time	Status	Database name
1331162434016	Storage Monitoring	2012-03-07 15:20:49.552	2012-03-07 15:22:35.804	Succeeded	GSDB-V10
1331171955983	Data Migration	2012-03-07 18:07:36.672	2012-03-07 18:07:38.134	Succeeded	GSDB-V10
1331171955983	Data Migration	2012-03-07 18:06:18.746	2012-03-07 18:07:29.892	Succeeded	GSDB-V10

1 - 3 of 3

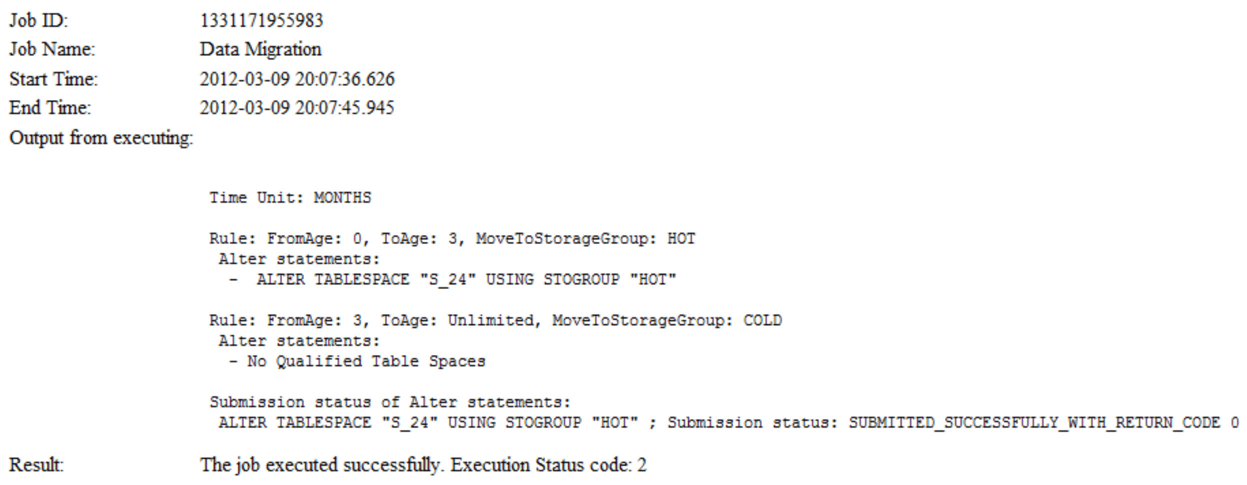
Define age ranges and specify the storage group to which to migrate table spaces that contain data within the age range. You cannot define overlapping age ranges.

Unit of time: Months

From Age	To Age	Storage Group
0	3	HOT
3	(unlimited)	COLD

Figure 11: Job history listing

The job history file reports the job details, age definition evaluation results, and SQL statements submitted as show in figure 12.



Job ID:	1331171955983
Job Name:	Data Migration
Start Time:	2012-03-09 20:07:36.626
End Time:	2012-03-09 20:07:45.945
Output from executing:	

```
Time Unit: MONTHS

Rule: FromAge: 0, ToAge: 3, MoveToStorageGroup: HOT
 Alter statements:
  -  ALTER TABLESPACE "S_24" USING STOGROUP "HOT"

Rule: FromAge: 3, ToAge: Unlimited, MoveToStorageGroup: COLD
 Alter statements:
  - No Qualified Table Spaces

Submission status of Alter statements:
 ALTER TABLESPACE "S_24" USING STOGROUP "HOT" ; Submission status: SUBMITTED_SUCCESSFULLY_WITH_RETURN_CODE 0
```

Result:	The job executed successfully. Execution Status code: 2

Figure 12: The SQL statements executed in the data migration job log.

Managing the impact of data movement

DB2 database administrators can limit the impact of a rebalancing operation by setting the UTIL_IMPACT_LIM database manager configuration parameter, and to prioritize which rebalance operations or other utilities are throttled through the UTIL_IMPACT_PRIORITY parameter. In DB2 V10.1, utility throttling was enhanced by introducing SUSPEND and RESUME options to the REBALANCE clause of the ALTER TABLESPACE SQL statement. These options have the advantage of being SQL statements rather than configuration parameters that require the use of CLP commands. In addition, these options offer the ability to halt a rebalancing operation completely during peak hours. This ability is useful in a multi-temperature environment in which entire table spaces are moved onto new sets of disks.

For example, suppose that you are moving last quarter's sales data from hot storage into a storage group associated with warm data. If your table space is called *sales_q1_2012* and your storage group is *sg_warm*, you do this by using the following statement:

```
ALTER TABLESPACE  sales_q1_2012 USING STOGROUP sg_warm

DB20000I  The SQL command completed successfully
```

This statement moves all data in *sales_q1_2012* onto the disks that back the storage group *sg_warm*. Talk to your storage administrator before you do this action if you use thin provisioning for disk resources, because the movement of the table space into this storage group might cause the utilization of the thin pools to rise quickly.

The MON_GET_REBALANCE_STATUS table function returns details on the active rebalance operations. The information from this table function can be used to monitor the progress and the status of all the rebalance operations.

```
SELECT varchar(tbsp_name, 20) AS tbsp_name, rebalancer_status,
rebalancer_extents_remaining AS extents_remaining,
rebalancer_extents_processed AS extents_processed FROM TABLE
(MON_GET_REBALANCE_STATUS(NULL, -1))

TBSP_NAME       STATUS  EXTENTS_REMAINING   EXTENTS_PROCESSED

-------------   ------  -------------------- --------------------

SALES_Q1_2012   ACTIVE                  6911                   34

 1 record(s) selected.
```

Suppose that your database experiences a heavy load, due to an influx of orders or a large reporting job required by senior management. Alternatively, suppose that you get a call from a panicked storage administrator, who warns you that the disk utilization on the disks that back the *sg_warm* storage group is rising too quickly, and might exhaust available storage before the storage staff can allocate more disks. In either case, you might want to suspend the rebalancer. To see all of the table spaces that are being rebalanced on any partition, and whose rebalancing operations should be suspended, issue the following query:

```
SELECT DISTINCT varchar(tbsp_name, 20) AS tbsp_name
FROM TABLE(MON_GET_REBALANCE_STATUS(NULL,-2)) AS t
WHERE rebalancer_status='ACTIVE'
 TBSP_NAME

 --------------------
```

```
SALES_Q1_2012

 1 record(s) selected.
```

Issue the following SQL statement to suspend this rebalancing operation:

```
      ALTER TABLESPACE  sales_q1_2012 REBALANCE SUSPEND

       DB20000I  The SQL command completed successfully.
```

This statement suspends the rebalancing operation of table space *sales_q1_2012* until you resume it. To see all suspended rebalancing operations on any database partition, issue the following query:

```
SELECT DISTINCT varchar(tbsp_name, 20) AS tbsp_name
FROM TABLE(MON_GET_REBALANCE_STATUS(NULL,-2)) AS t
WHERE rebalancer_status='SUSPENDED'

TBSP_NAME

--------------------

 SALES_Q1_2012

1 record(s) selected.
```

This shows that the rebalancing of table space *sales_q1_2012* is now suspended. To resume rebalancing, issue the following query:

```
      ALTER TABLESPACE sales_q1_2012 REBALANCE RESUME

      DB20000I  The SQL command completed successfully.
```

Upgrading an existing database into a multiple temperature storage database

Before you upgrade an existing database into a multi-temperature storage database, it must be first upgraded to DB2 V10.1. Backup the database before you upgrade the DB2 server and databases. Refer to the Upgrading databases section in DB2 V10.1 documentation for complete details. The following procedure describes the sequence of steps for upgrading an existing database into a multi-temperature storage database.

1. Upgrade the existing database to DB2 V10.1 version by using the UPGRADE DATABASE command.

   If your database was enabled for automatic storage before the upgrade, you have a default storage group named IBMSTOGROUP that exists in the SYSCAT.SYSGROUPS table. All your automatic storage managed table spaces are in this default storage group.

2. Identify the number of storage groups, the storage paths to add to them and their storage attributes. Create the identified storage groups.

3. Identify the automatic storage table spaces that you want to reside in your tiered storage groups created in step 2. These table spaces are typically associated with date-based partitioned data tables. If there are no table spaces identified in this step, skip to step 5.

   ```
   SELECT TBSP_NAME, TBSP_CONTENT_TYPE
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
      ORDER BY TBSP_ID
   ```

4. Move the table spaces identified in step 3 to their target storage group.

   If you have many table spaces to process, take care to manage the number of active rebalances occurring at one time to avoid performance impact to your database. To see how to query the system for the number of active rebalances and how to manage their execution, see the "Using rebalance suspend/resume to manage the impact of data movement" section. Moving your most performance critical table spaces first helps you take advantage of your multi-temperature storage configuration more quickly.

5. Identify the database managed table spaces that you want to convert to automatic storage managed table spaces and reside in the tiered storage groups created in step 2. If there are no table spaces identified in this step, skip to step 7.

6. Alter the table spaces identified in step 5 by specifying the MANAGED BY AUTOMATIC STORAGE and the USING STOGROUP options.

   When you convert a DMS table space to be automatic storage managed, you must issue an explicit ALTER TABLESPACE REBALANCE command to initiate the process of moving the data into the target storage group. The REBALANCE command begins after the transaction is committed.

7. Formulate a plan on how you want to define your policies for moving data between your configured storage groups. You can then create a recurring job in Optim Configuration Manager by using this data to set up your automatic data movement polices.

Using data tags and DB2 workload manager to prioritize activities based on the data that is being accessed

The typical and recommended approach to managing work in a DB2 warehouse environment can be found in *DB2 best practices: Implementing DB2 workload management in a data warehouse*. This best practices article is available at IBM developerWorks®: **http://www.ibm.com/developerworks/data/bestpractices/workloadmanagement/**
This paper is referenced more than once in this section.

Using the approach described in the DB2 workload management (WLM) best practices article, the overall health of the system can be enhanced by controlling the mix and volume of work on the system at any one time. With a multi-temperature system, this approach can continue to be sufficient, especially if it is the larger, more resource-intensive queries that work on colder data so that cold data work would naturally be limited by the standard approach. However, in some cases, the queries that touch colder data might be intermingled with those touching warmer data so there is not a natural division within the standard approach. In such environments, if the performance objectives require that controls be placed on the queries that touch colder data in order to protect resources for work on warmer data, then additional workload management configuration changes are needed. Using some of the existing DB2 WLM features in combination with the new V10.1 data tags and some of the new DB2 WLM features, you can combine data temperature with the type and cost of the work to determine what resources are to be assigned to that work.

A data tag is a numeric identifier that can be assigned to a table space or storage group. Use data tags to assign an identity to data contained within that table space. The data tag for a table space can be either specified directly, or inherited from its storage group. Using DB2 WLM, you can identify which table spaces are being accessed by an activity and prioritize the work that is running on your system based on the data that is accessed. This can be done before the activity begins to run (predictive), while the activity is running (reactive), or both.

The meaning of the data tag identifier is up to you and depends on how you use it with the DB2 WLM controls. There are two intended use cases for data tags.

- To identify and prioritize work based on the temperature (hot versus cold) of the data the work touches.
- To allow a more general identification of the data that is being touched by the work. In this case, the data tag does not represent the temperature of the data that is being accessed. Instead, it might be used in a data-centric scenario where, for example, the data that is being touched represents the part of the organization you come from and perhaps the priority of the work.

Figure 13 shows a revised or tuned (stage 2) version of the DB2 WLM template that originated from the DB2 WLM best practices paper. In this diagram, all user work is mapped to a user-defined service superclass. In the service superclass, there are several service subclasses, representing different types and sizes of queries based on their estimated cost. A work action set is used to map the appropriate queries (based on size and type) to the appropriate service subclass within the service superclass. In addition, there are currently two concurrency thresholds to limit the number of LOAD activities to 1 at a time and the number of complex activities to five.

We use this DB2 WLM setup as our base example to add on to for the next two subsections. Any DDL shown assumes that the same names for the existing service classes, work action sets, and work class sets as those in the template script that is provided in the DB2 WLM best practices.

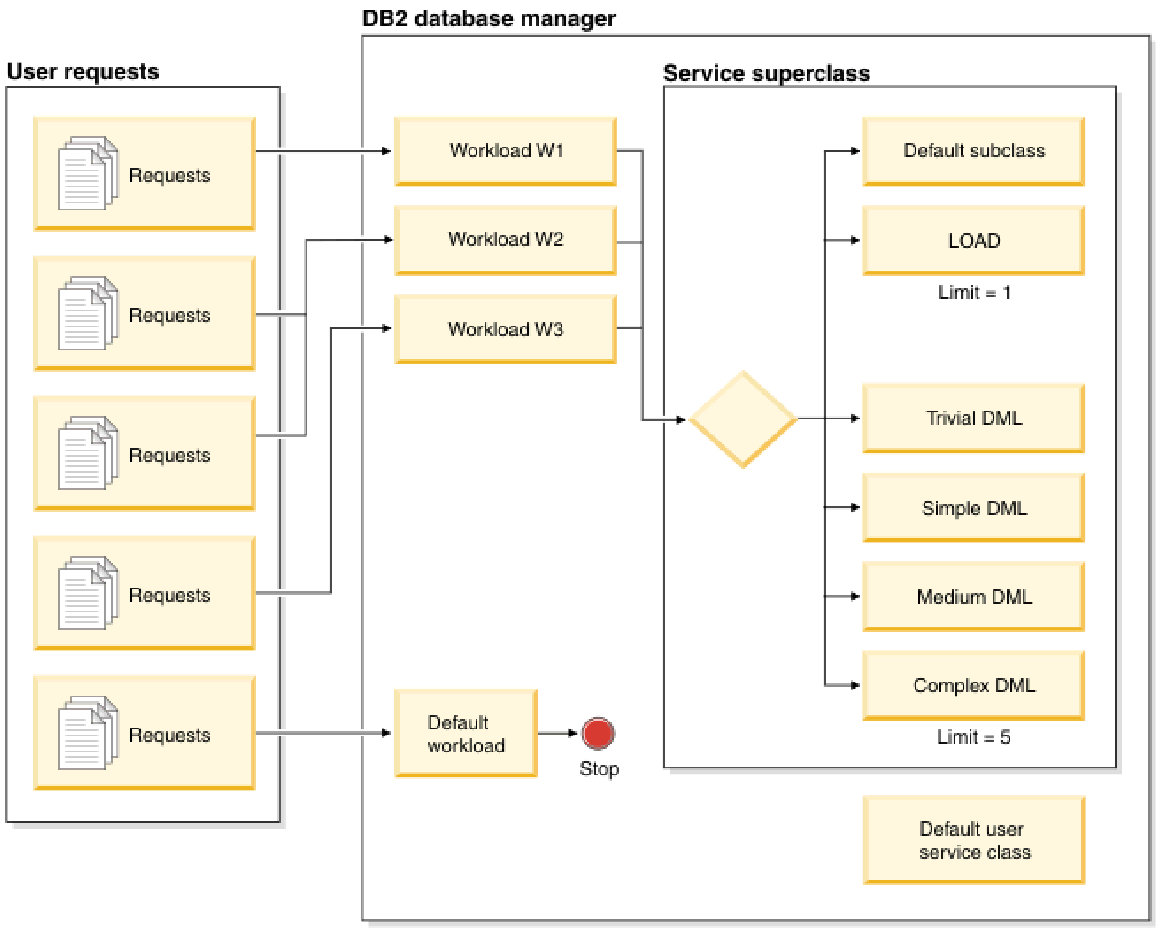


Figure 13: Tuned "Best Practices" WLM Configuration.

Using data tags with DB2 WLM predictively

As previously mentioned, in a multi-temperature system, the approach described in the DB2 WLM best practices where work is isolated and treated differently based on the activity type and cost (timerons) might be sufficient. But, there might be situations where you would also like to treat work differently based on the data that might be touched. For example, your workload might contain some complex (based on cost) queries that touch both hot and cold data and you would like to give these complex queries that touch the hot data more system resources than those touching the cold data.

Data tags can be used by DB2 WLM through work class sets and work action sets to predictively treat work differently based on the data the work is believed or expected to access. At compile time, the DB2 optimizer provides an estimated data tag list, similar to cost (timerons) and cardinality estimates. The estimated data tag list contains the data tag values for all the table spaces that the compiler believes will be accessed during the execution of the activity. You can now define work classes to identify activities that have a specific data tag in the list of estimated data tags provided by the compiler and then dictate, by using work action sets, what action to apply to those activities. An example of one of the more useful and popular work action types is MAP ACTIVITY, which you can use to map activities to different service subclasses. It is this work action type that is used in this example to map activities to different priority service subclasses depending on the type and size of the activity and also by the "priority" of the data that is believed to be accessed by the activity.

To better explain, we go through an example, based on the revised template in the DB2 WLM best practices as shown in Figure 13.

Assume that storage group SGHOT contains one table space, called TSHOT. Table space TSHOT has the following properties:

- The table space contains new data that is less than six months old.
- The table space is stored on a fast storage device.
- Work that touches data in the table space is considered high priority work.

Storage group SGCOLD contains a table space TSCOLD. Table space TSCOLD has the following properties:

- The table space contains data that is older than six months old
- The table space is stored on a slower storage device.
- Work that touches data in the table sapce is considered lower priority work.

Each storage group is assigned a data tag identifier and each table space within the storage groups inherit their data tag values from their parent storage group. The data tag identifies the data that is touched by activities. You decide the meaning of the data tag. In this example, data tag 1 is assigned to TSHOT and 5 is assigned to TSCOLD.

```
ALTER STOGROUP SGHOT DATA TAG 1

ALTER STOGROUP SGCOLD DATA TAG 5
```

If you plan to use the data tag feature with WLM in a multi-temperature environment, apply the data tags at the storage group level to allow the table spaces to inherit their data tag values from their parent storage group. This way, if the table space is moved from one storage group to another, it automatically inherits its new data tag value from

the new storage group and the WLM priority is adjusted automatically, saving you from adjusting the data tag values manually.

As part of this example, assume that we ignore the temperature of the data that the trivial and simple DML activities touch because their impact to the system performance is inconsequential. But, the medium and complex DML activities are larger and they sometimes touch the newer, hotter data. These activities also sometimes touch the older, colder data. Therefore you want to separate out the medium and complex queries that touch the colder data and give them less system resources as shown in the following diagram.

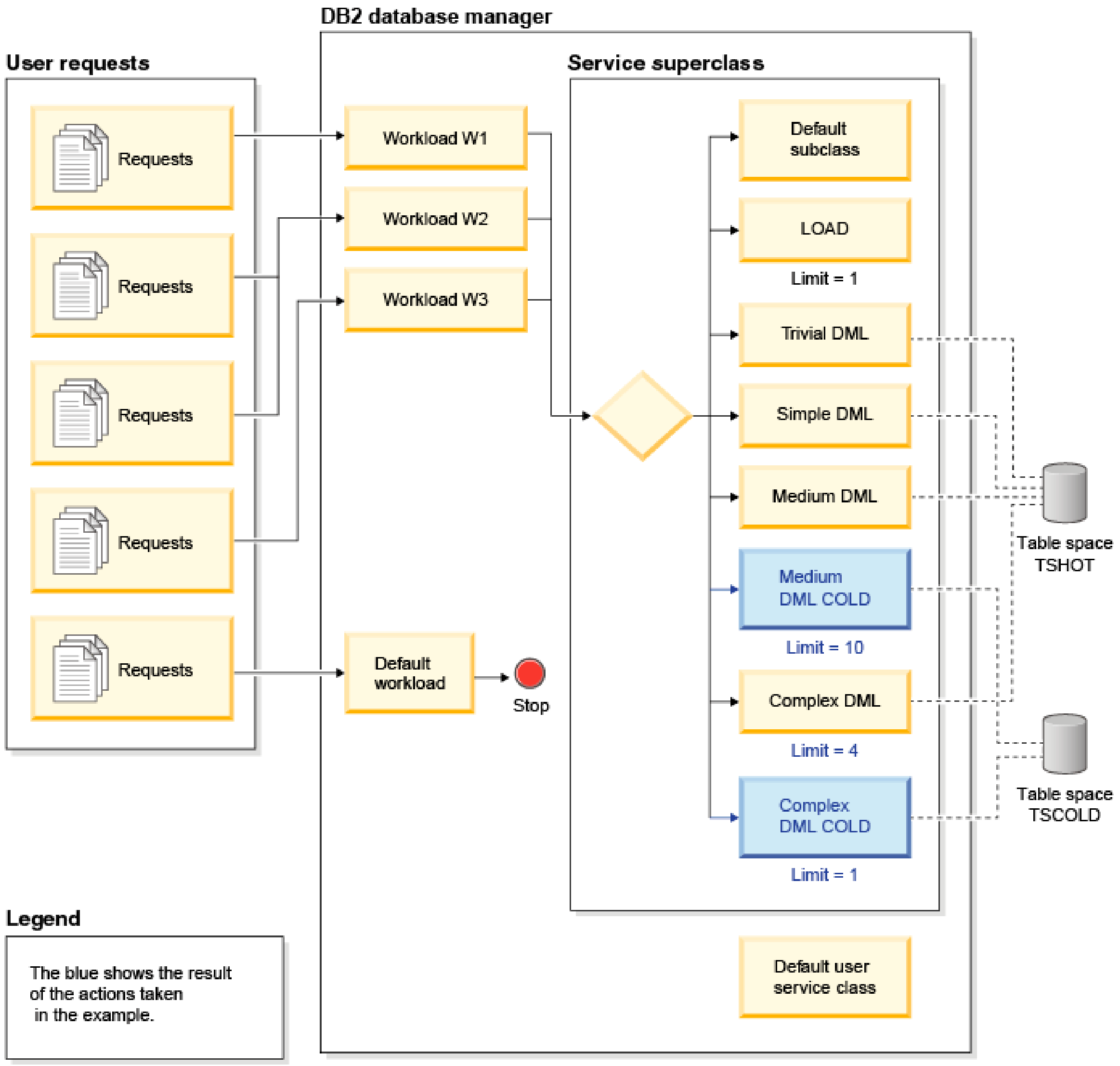


Figure 14: Revised DB2 WLM best practices template that uses data tags predictively

To achieve this setup, two new service subclasses are created, one representing medium DML activities that touch low priority data and one representing complex DML activities that touch low priority data. In addition, two new concurrency thresholds are applied to the service subclasses so that 10 medium sized queries that are predicted to touch the older, cold data are allowed to run at a time and only one complex query that is predicted to touch the colder data can run at a time. In addition, the existing threshold that allowed five complex queries to run concurrently is changed so that only four queries can now run at a time.

Before this change, only five complex queries could run concurrently. We want to maintain this restriction, so we split the number of complex queries that touch hot data and those touching cold data so that they still only totaled five concurrent queries. For medium queries, there was previously no concurrency limit applied so only those queries that touch the cold data are limited.

```
CREATE SERVICE CLASS MEDIUM_DML_COLD UNDER WLMBP_MASTER

CREATE SERVICE CLASS COMPLEX_DML_COLD UNDER WLMBP_MASTER

CREATE THRESHOLD "WLMBP_MEDIUM_DML_COLD_CONCURRENCY"

        FOR SERVICE CLASS "MEDIUM_DML_COLD" UNDER "WLMBP_MASTER"

        ACTIVITIES ENFORCEMENT DATABASE

        WHEN CONCURRENTDBCOORDACTIVITIES > 10 AND
QUEUEDACTIVITIES

        UNBOUNDED  CONTINUE

CREATE THRESHOLD "WLMBP_COMPLEX_DML_COLD_CONCURRENCY"

        FOR SERVICE CLASS "COMPLEX_DML_COLD" UNDER "WLMBP_MASTER"

        ACTIVITIES ENFORCEMENT DATABASE

        WHEN CONCURRENTDBCOORDACTIVITIES > 1 AND QUEUEDACTIVITIES

        UNBOUNDED  CONTINUE

ALTER THRESHOLD “WLMBP_COMPLEX_DML_CONCURRENCY”

         WHEN CONCURRENTDBCOORDACTIVITIES > 4 AND
QUEUEDACTIVITIES

          UNBOUNDED CONTINUE
```

In order to isolate out the lower priority medium and complex DML activities that touch the cold data from those activities that do not, we need to update the already existing work class set to add in two new work classes (for simplicity, we assume that medium and complex activities have the same estimated cost range as what is in the DB2 WLM best practices article). These new work classes are positioned ahead of the MED_COST_DML and COMPLEX_COST_DML work classes so that if the activity in the medium or complex cost range has a different data tag other than 5 (representing old, cold data) or if it has no data tag, it falls into the MEDIUM_COST_DML or COMPLEX_COST_DML work classes.

```
ALTER WORK CLASS SET WLMBP_WORK_CLASSES

      ADD WORK CLASS MEDIUM_COST_DML_COLD WORK TYPE DML

            FOR TIMERONCOST FROM 300000 TO  5000000 TIMERONS

            DATA TAG LIST CONTAINS 5

            POSITION BEFORE MEDIUM_COST_DML

      ADD WORK CLASS COMPLEX_COST_DML_COLD WORK TYPE DML

            FOR TIMERONCOST FROM 5000000 TO UNBOUNDED

            DATA TAG LIST CONTAINS 5

            POSITION BEFORE COMPLEX_COST_DML
```

Next, alter the existing work action set and add two new mapping work actions and apply them to the two new work classes to map the medium and complex activities to the appropriate service class based on the data they touch:

```
ALTER WORK ACTION SET WLMBP_WORK_ACTIONS

      ADD WORK ACTION MAP_MEDIUM_COST_DML_COLD

            ON WORK CLASS MEDIUM_COST_DML_COLD

            MAP ACTIVITY TO MEDIUM_DML_COLD

      ADD WORK ACTION MAP_COMPLEX_COST_DML_COLD

            ON WORK CLASS COMPLEX_COST_DML_COLD

            MAP ACTIVITY TO COMPLEX_DML_COLD
```

Now, medium and complex activities that are expected to touch older, cold data are throttled back based on the concurrency thresholds values applied, which allows activities that are expected to not touch the colder data to run with more system resources.

Using Data Tags with DB2 WLM reactively

There are times when the compiler cannot accurately estimate which table space a particular query might touch. For example, running a query against a partitioned table that uses parameter markers. In this case, the compiler cannot necessarily determine in advance what table ranges will be accessed. This case is when reactive prioritization can be helpful.

The new DB2 V10.1 in-service-class DATATAGINSC threshold can be used to perform an action on work based on the data accessed at run time. Using this threshold, you can perform the standard actions supported by most DB2 WLM thresholds, such as continue, stop execution, and notification by collecting data. In addition, by using this threshold, you can remap an activity to a different service subclass within the same service superclass as it accesses data from a table space with a particular data tag. For example, you can define one of these new thresholds to remap any activity that runs in a subclass that touches a table space with a data tag of 5 to a different service subclass.

Typically, the remap action supported by any of the DB2 WLM thresholds is used for one of the following purposes:

- Monitoring. For example, when an event occurs that causes the remap action to occur, the activity is remapped to a service subclass that is configured to capture metrics and activity information.

- Control: To reactively change the system resources allocated to work based on an event (such as when work touches cold data).

If you plan to use the remap action for control purposes, use it in conjunction with either OS WLM or with the new V10.1 DB2 workload management dispatcher. For more information about the remap action, see *DB2 best practices: Implementing DB2 workload management in a data warehouse* at IBM developerWorks.

This example goes through a scenario that uses the new DATATAGINSC threshold in combination with the new DB2 workload management dispatcher to reactively control the resources allocated to work based on the data it touches. This example is based on the revised template from the DB2 WLM best practices as shown in Figure 13 where there is a work action set that maps activities to the different service subclasses based on the type and size of the work.

In this example, ignore the temperature of the data the trivial and simple DML activities touch since their impact to the system is inconsequential. However, the medium and complex queries can affect the system. They contain parameter markers and depending on their values, they can touch both the newer, hotter data; the older, colder data; or both. The kind of data touched can be determined only at run time. If the medium and complex queries touch the older, colder data when they run, you want to give them less system resource by remapping them to a lower priority service subclass.

The following diagram describes this situation.

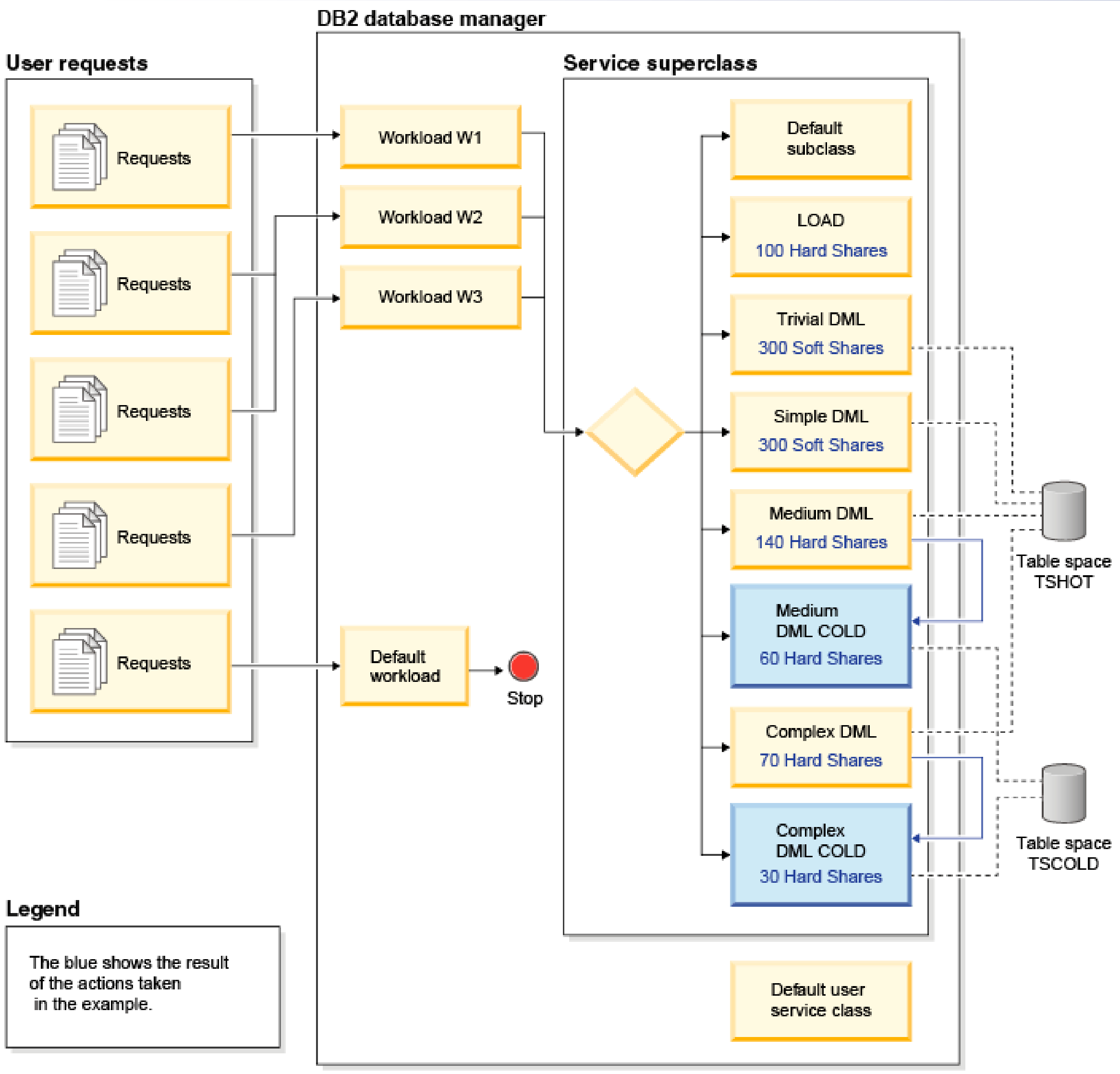


Figure 15: Revised DB2 WLM best practices template that uses data tags reactively with workload management dispatcher

Because the medium and complex queries might touch both new and old data, two new service subclasses (Medium DML COLD and Complex DML COLD) are created the same way they were in the previous example.

When you use the DB2 workload management dispatcher, typically, soft shares are used to allow higher priority work to utilize any spare capacity available on the system while hard shares are used to ensure that higher impact work is always strictly limited in the presence of other work to not degrade the performance of the higher priority work

In this example, assume that you set up the service subclasses so that the amount of system resources (CPU) is controlled by using some of the new DB2 workload management dispatcher features: soft and hard shares. The trivial and simple queries are given the most processor resources, 300 soft shares each. Because they are soft shares, if there are any additional processor resources that are not being used, they can also tap into those resources. Load activities are given 100 hard shares, medium activities not touching the cold data are given 140 hard shares, but they get 60 hard shares only if they do end up touching the cold data at any time during their execution. Complex queries not touching cold data are given 70 hard shares, but they are given only 30 hard shares if they touch the cold data at any time during their execution.

Activities running in the service subclasses that have hard shares applied to them can tap only into unused processor resources from the trivial and simple subclasses (soft shares) when there is nothing that is running in them. So, for this example, load activities, and medium and complex queries can tap into unused processor resources only if there are no trivial and no simple queries that are running.

The following example shows only the DDL to apply the soft shares to the trivial service subclass and the hard shares to the medium service subclass. Applying the soft and hard shares to the other subclasses is similar.

```
ALTER SERVICE CLASS WLMBP_TRIVIAL_DML UNDER WLMBP_MASTER

      SOFT CPU SHARES 300

 ALTER SERVICE CLASS WLMBP_MEDIUM_DML UNDER WLMBP_MASTER

      HARD CPU SHARES 140
```

In this scenario, instead of using a work action set to predictively map activities to different priority service subclasses based on the data the activities are expected to touch, all of the medium activities are initially mapped to the "Medium DML" service subclass and all of the complex activities are initially mapped to the "Complex DML" service subclass. As the medium and complex activities run, if they touch the older, cold data, they are bumped down to the appropriate lower priority service subclass (based on processor shares allocated). This is achieved by using the DATATAGINSC threshold with the remap action:

```
   CREATE THRESHOLD WLMBP_REMAP_MEDIUM_TO_MEDCOLD

        FOR SERVICE CLASS WLMBP_MEDIUM_DML

        UNDER WLMBP_MASTER ACTIVITIES

        ENFORCEMENT MEMBER

        WHEN DATATAGINSC IN (5)

        REMAP ACTIVITY TO WLMBP_MEDIUM_DML_COLD


  CREATE THRESHOLD WLMBP_REMAP_COMPLEX_TO_COMPLEXCOLD
```

```
FOR SERVICE CLASS WLMBP_COMPLEX_DML

UNDER WLMBP_MASTER ACTIVITIES

ENFORCEMENT MEMBER

WHEN DATATAGINSC IN (5)

REMAP ACTIVITY TO WLMBP_COMPLEX_DML_COLD
```

In this example, any medium sized activities that touch the older, cold data (data in a table space with a data tag of 5) are remapped to the "Medium DML COLD" service subclass and therefore might get less processor resource. Any complex activities that touch the older, cold data (data in a table space with a data tag of 5) are remapped to the "Complex DML COLD" service subclass, possibly getting even less processor resource.

Backup and recovery considerations

When a data warehouse includes large volumes of cold data on inexpensive, slower storage devices, it takes additional time to run a full database backup compared to an environment that contains only fast storage devices. Backup time is directly related to the speed of the storage devices. Therefore for warehouses with multi-temperature data, the best practice recommendation is to implement online table space backup. Use a backup strategy based on table space backups rather than full database backups so that you can take granular backups based on the temperature of the data. For example, you might back up cold and dormant data once a month (synchronized with the movement of data from one temperature tier to another), and you might back up hot and warm data on a daily basis.

Use table space backups rather than full database backups so that you can back up more volatile hot and warm data more frequently and mostly static cold and dormant data less frequently.

Recovery methods depend on the symptoms of the problem that requires you to restore the data (for example, a table space loss, a catalog partition loss, or a database partition loss).

For more information, see the best practices paper on backup and recovery that is available from the following site:
http://www.ibm.com/developerworks/data/bestpractices/isasrecovery/index.html

Setting up an alternative copy of your database

At times database administrators need to set up multiple copies of the same database on different systems. One such scenario would be setting up the same copy of the database on multiple QA and sandbox systems for rigorous testing of new functionality or application changes before they can be certified to roll out onto the production systems.

Not all systems where the database needs to be set up have the same kinds of devices, storage tiers, or paths. In such situations, a redirected restore operation can be used to set up alternative copies of the same database on different locations. DB2 V10.1 provides a new SET STOGROUP PATHS command to redirect storage paths for each storage group in the backup image.

Consider the following example where the inventory database has four storage groups *ibmstogroup*, *sg_hot*, *sg_warm*, *sg_cold*, and a DMS table space *dms_tbs1*. The sequence of steps to set up an alternative copy of the inventory database as *inventory_qa* would be as follows.

1. Make sure the new database path and storage paths that you use exist on your host and are available for the DB2 database to use.

2. The redirected restore command sets up the database path in `/qa15/db2/db2dir` and rename the database to *inventory_qa*:

```
RESTORE DATABASE inventory FROM /database_backup/INVENTORY
DBPATH ON  '/qa15/db2/dbdir' into inventory_qa REDIRECT
WITHOUT PROMPTING

SQL1277W  A redirected restore operation is being
performed. During a table space restore, only table spaces
being restored can have their paths reconfigured. During a
database restore, storage group storage paths and DMS table
space containers can be reconfigured.

DB20000I  The RESTORE DATABASE command completed
successfully.
```

3. Assign the default storage group, *ibmstogroup*, storage paths.

```
SET STOGROUP PATHS FOR ibmstogroup ON
'/qa15/db2/default_path1',  '/qa15/db2/default_path2',
'/qa15/db2/default_path3'

DB20000I  The SET STOGROUP PATHS command completed
successfully.
```

4. Assign the rest of the storage group storage paths to share the one tier of storage on the QA system.

```
SET STOGROUP PATHS FOR sg_hot ON   '/qa15/db2/data_path1',
'/qa15/db2/data_path2'. '/qa15/db2/default_path3'

DB20000I  The SET STOGROUP PATHS command completed
successfully.

SET STOGROUP PATHS FOR sg_warm ON   '/qa15/db2/data_path1',
'/qa15/db2/data_path2', '/qa15/db2/default_path3'

 DB20000I  The SET STOGROUP PATHS command completed
successfully.
```

```
SET STOGROUP PATHS FOR sg_cold ON '/qa15/db2/data_path1',
'/qa15/db2/data_path2', '/qa15/db2/default_path3'

DB20000I  The SET STOGROUP PATHS command completed
successfully.
```

5. Assign the container for the DMS table space to their new locations. In this example, table space *dms_tbs1* has a table space identifier of 5.

```
SET TABLESPACE CONTAINERS FOR  5 USING  (PATH
'/qa15/db2/dms_storage/INVENTORY_QA/dms_tbs1')

DB20000I  The SET TABLESPACE CONTAINERS command completed
successfully.
```

6. Proceed with the redirected restore

```
RESTORE DATABASE inventory CONTINUE

DB20000I  The RESTORE DATABASE command completed
successfully.
```

7. Modify the storage group media attributes to reflect the performance of the QA system.

```
ALTER STOGROUP sg_cold READ DEVICE RATE 100 OVERHEAD 6.725

DB20000I  The SQL command completed successfully.

ALTER STOGROUP sg_warm READ DEVICE RATE 100 OVERHEAD 6.725

DB20000I  The SQL command completed successfully.

ALTER STOGROUP sg_hot READ DEVICE RATE 100 OVERHEAD 6.725

DB20000I  The SQL command completed successfully.
```

Appendix A: Determining media attributes by using the static approach

In order to tune the media attributes, you must know about the physical devices that back the storage paths. To gather information about the disks used by the DB2 instance:

- Determine which file systems are used as database storage paths.
- Determine the mapping between those file systems and the disks or LUNs seen by the host systems.
- Query the disk devices to discover the serial number of those LUNs from the host operating system.
- Using the storage controller management software and those LUN serial numbers, determine the actual physical disk (group of disks), at the back-end array.
- Compute the storage path media attributes by using the information about the back-end physical disk

Determining the file systems used as storage paths in a database

The first step in the process of mapping host addressable devices to the actual physical disks is to learn what file systems are used to place table space containers – that is, automatic storage (AS) storage paths used by the database. This information can be easily queried by using the ADMIN_LIST_DB_PATHS table function:

```
select distinct PATH from TABLE(ADMIN_LIST_DB_PATHS())where
TYPE='DB_STORAGE_PATH'

PATH
-----------------------------------------------------------------
/db2/hot01
/db2/warm01
/db2/warm02
/db2/warm03
/db1/db2inst1


5 record(s) selected.
```

Identifying the physical disk drives by using the serial numbers queried from host side:

For platform-specific instructions on how to obtain the device serial number, see Appendix B for Linux platforms and Appendix C for AIX platforms.

When you have the serial numbers of the LUNs, you can query the host-to-LUN-mappings defined at the storage controller to determine from which physical disk arrays your file systems use.

How to query the storage controller for this information is dependent on the type of storage you have. A few examples include:

- IBM DS5000 Series controllers: IBM DS Storage Manager Client GUI.
- IBM DS8000 series controllers: IBM System Storage Management Console (by using IBM Tivoli Productivity Center GUI) or by command-line interface (DSCLI).
- EMC Symmetrix Series: EMC Symmetrix Management Console (SMC) web interface or Symmetrix Solutions Enabler command-line tools (SYMCLI)
- NetApp FAS Series iSCSI Controllers: NetApps OnCommand tools

A disk array is defined over a number of disks of the same type, which is called a disk group. The host addressable device (or hdisk) is a logical slice (LUN) from such disk array.

For example, a RAID 5 (3+1) array has four physical hard disks from same device type (from a single disk group in the back-end), such as 450GB 15K FC drives (which are Seagate model ST345085). You can now obtain the disk drive spec data sheet by doing a lookup of the model number from the disk drive manufacturer's website.

Computing DEVICE READ RATE or TRANSFERRATE and OVERHEAD attribute values

When you have the disk drive spec sheet, you can compute the required media attributes easily for the CREATE STORGROUP command.

Going back to the example of a 450GB 15K Seagate ST345085 FC drive, from the spec sheet we can find the following specific device details:

- Spindle speed: 15K RPM
- Seek Time (Avg Read): 3.4 ms
- Transfer Rate (Sustained): 122 to 204 MB/s

If you configure table spaces to inherit the media attributes, then the storage group attribute DEVICE READ RATE can be computed as follows:

DEVICE READ RATE = Average transfer rate from the spec data sheet = (204 + 122) / 2 = 163 MB/s

If you are not directly setting or overriding the storage group media attribute at the table space level, then the TRANSFERRATE attribute can be calculated by using the following formula:

TRANSFERRATE = (1 / DEVICE READ RATE) × 1000 / 1024000 × page size

Using the average transfer rate from the spec data sheet, the TRANSFERRATE attribute for a table space with a 16K page size (16384 bytes) that uses a LUN from this device can be computed as follows:

TRANSFERRATE = (1 / 163) × 1000 / 1024000 × 16384 = 0.098

The OVERHEAD attribute value is computed as follows:

OVERHEAD = average seek time in milliseconds + (0.5 * rotational latency)

where:

- 0.5 represents the average overhead of one half rotation
- Rotational latency (in milliseconds) is calculated for each full rotation, as follows:

  (1 / rpm) × 60 × 1000

For our 450GB 15K Seagate ST345085 FC drive, the OVERHEAD attribute is as follows:

rotational latency = (1 / 15000) * 60 * 1000 = 4

OVERHEAD = 3.4 + (0.5 * 4) = 5.4

For more details on these computations, see DB2 information center: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.perf.doc/doc/c0005051.html

Appendix B: Determining a device serial number on Linux systems

Mapping the storage paths to LUNs (host addressable disk device)

After we know the storage paths that the database uses, we can map those file systems to a disk or set of disks (LUNs) seen by the host.

Assume that you identified `/db1/db2inst1` as the storage path used by the database with the `ADMIN_LIST_DB_PATHS` table function. In that case, you can locate the LUN (hdisk) where this particular file system is created by using the following commands:

```
[root]# mount | grep -i db1

/dev/mapper/vg_svtlnx26-LogVol00 on /db1 type ext3 (rw)

[root]# lvdisplay -m /dev/mapper/vg_svtlnx26-LogVol00

--- Logical volume ---

LV Name /dev/vg_svtlnx26/LogVol00

VG Name vg_svtlnx26

.

.

.

Physical volume /dev/sda5

Physical extents 12000 to 23848
```

Now that we know which hdisks are used by the database as storage paths, we can trace them back to the storage controller array.

Finding the serial numbers of the hdisks (LUNs)

Assume that we have file system `/db1`, which is used by the database as a storage path, and is created by using device (LUN) hdisk1.

The device-specific VPD page can be determined by using the **sg_inq** command. This command requires the host to have SG utilities RPM files installed. Assuming that `/db1` is on a device mapper multipath (DM MP) device:

```
[root]# sg_inq -p=0x80 /dev/dm-12

VPD INQUIRY: Unit serial number page

Unit serial number: 900547065000
```

Appendix C: Determining a device serial number on AIX systems

Mapping the storage paths to LUNs (host addressable disk device)

After we know which storage paths are used by the database, we can easily map those file systems to a disk or set of disks (LUNs) seen by the host.

Assume that you identified `/db1/db2inst1` as the storage path used by the database with the earlier table function,. In that case, you can locate the LUN (hdisk) where this particular file system is created by using the following steps.

```
root> mount | grep -i /db1

/dev/'''db1lv''' /db1 jfs2 Aug 28 13:07 rw,log=/dev/loglv01

root> lslv -l db1lv

db1lv:/db1

PV COPIES IN BAND DISTRIBUTION

hdisk1 798:000:000 20% 160:160:159:160:159
```

Now that we know which hdisks are used by the database as storage paths, we can trace them back to the storage controller array.

Finding the serial numbers of the hdisks (LUNs)

Assume that we have file system `/db1`, which is used by the database as a storage path, and is created by using device (LUN) hdisk1. You can get vital product data (or VPD) information of a device by using the **lscfg** command.

```
root> lscfg -vl hdisk1

hdisk1           U5791.001.9920MAY-P2-C02-T1-
W50050768014036DC-L0  MPIO FC 2145

Manufacturer................IBM

Machine Type and Model......2145

ROS Level and ID............0000

Device Specific.(Z0)........0000043268101002

Device Specific.(Z1)........020060c

Serial Number...............600507680183006EC80000000000006F
```

Appendix D: Determining media attributes by using the dynamic approach

Using I/O performance monitoring tools to gauge disk throughput

The previous methodology disregarded the external storage characteristics and treated those devices as internal disks and used static (vendor published) specs to compute media attributes. Another approach is to monitor the I/O characteristics of the disks that back the DB2 automatic storage paths and use the results to estimate a possible value.

Disk I/O performance monitoring can be achieved by using the tools provided by your storage vendors or by using a free tool such as iometer.

Regardless of the monitoring tool used, you must pay attention to the following factors to make a reasonably accurate estimation:

The DB2 optimizer is a complex tool that takes in large number of variables into consideration before it creates access plans and chooses the optimal plan. Therefore, the goal of the I/O performance is not to compute an exact number, but rather understand the scale of difference (or variance) of the observations – for example, a TRANSFERRATE attribute change from 0.065 to 0.091 is less likely to result in different access plans than a change from 0.065 to 0.009.

- In general, any estimation exercise depends on having a large sample size (or data set) and a reasonably high sampling frequency to improve the quality of the estimation.

- Select a good representative period from each day to monitor your workload I/O profile. For example, monitoring on a single day where the month-end queries (or batch jobs) are run would not yield a good representative sample of stable production I/O characteristics.

- Be cognizant of anomalies in your observations, such as sudden dips or relatively short spikes in I/O throughput. For example, if a read-only query is being run on a data set which has a high locality of reference (which does not necessarily mean adjacent rows, but how the data blocks are located on disk), the storage controller prefetcher might kick in. Depending on how much controller cache you have and how much cache is available at that point in time, DB2 I/O might be mainly hitting this high-speed cache which might show a spike in I/O throughput observations. Another example is a scenario where a snapshot copy or a disk clone process for disaster recovery purposes is kicked off while a set of tables is getting updated frequently. Storage controllers typically use Copy on

Write (COW), which would mean that, before updating the data blocks from update/insert SQL, the storage controller needs to update the clone target volumes. This situation has a negative impact on I/O during that period and can show up as a dip in the observed throughput.

- Know where your disks are coming from – observations from directly attached storage vs. SAN storage have different factors that can skew data throughput. For example, your storage can be connected through two Fibre Channel ports into the SAN fabric, while 20 hosts are accessing LUNs from that particular controller. In such a configuration, heavy I/O traffic generated from hosts outside of DB2 system can skew your observations.

In summary, you should have a good idea about the context of the observed throughput values and not consider them as precise values that can be directly plugged in without due diligence.

After your data is collected and analyzed, some form of a general trend analysis might be required to identify infrequent changes or anomalies in the workload, and normalize those observations. If your database workload is predictable, a simple average might be good enough. The value you calculate is equivalent to the transfer rate value specified by the disk drive manufacturer.

However, another point to keep in mind is that you must still calculate the overhead by using the published spec sheet for the disk drive. All current tools report only the I/O throughput seen from the host that runs the monitoring software.

Tools that are supplied by storage controller vendors usually can query the controller itself and look at metrics such as controller cache hit/miss ratios or disk array utilization. If your workload is random in nature and difficult to do any reasonable level or trending, you might want to introduce additional factors, such as I/O wait times, and controller information reported by a vendor-specific tool to bring in more evidence to understand the I/O characteristics. If the workload I/O is near impossible to trend, using static values is a safer approach. However, a reasonable variance in the observations is acceptable, and the key is to understand the range rather than focusing on one perfect value.

Conclusion

This paper describes how to design, implement, and manage a multi-temperature data warehouse. It also describes the many advantages of multi-temperature data warehouses. They can help reduce your costs by using inexpensive storage for the data you rarely need to access. They enable you to keep all of your data accessible within a single data warehouse, even as the volume of historical data grows significantly over time. In addition, with the help of DB2 workload management, you can allocate more resources to high-priority requests for current data than to low-priority requests for extremely old data.

Summary of key recommendations

- To help ensure predictable performance, all the paths that you assign to a storage group should have the same media characteristics: latency, device read rate, and size.
- For best query performance, set the storage group media device attributes OVERHEAD and DEVICE_READ_RATE to reflect the storage path capabilities.
- Physically separate hot, warm, cold, and dormant data by storing data in table spaces based on the temperature of the data. Store your hot data on fastest storage, warm data on fast storage and store cold and dormant data on slower storage.
- Use table partitioning and arrange table partitions in table spaces so that the granularity of the data stored in each table space matches your schedule for moving data from one temperature tier to another.
- If you have multiple tables that share a table space, ensure that they have the same temperature characteristics. All the data in a table space is moved when its storage group is altered.
- When an automatic storage table space inherits the TRANSFERRATE attrbute setting from the storage group it is using, the DEVICE READ RATE attribute value of the storage group, which is in megabytes per second, is converted into milliseconds per page read accounting for the PAGESIZE attribute setting of the table space.
- Be careful to not overload your system by moving many table spaces at one time. Each ALTER TABLESPACE … USING STOGROUP … statement initiates an implicit background REBALANCE operation at commit time to move the table space data to the specified storage group.
- Use table space backups rather than full database backups so that you can back up more volatile hot and warm data more frequently and mostly static cold and dormant data less frequently.
- When using data tags with WLM, consider setting the data tag values at the storage group level, allowing the table spaces to inherit their values from their parent storage group. Doing so might save you from having to update the data tag values for the table space if you move it from one storage group to another.
- If you plan to use the remap action for control purposes through any of the DB2 WLM thresholds that support remap, use it in conjunction with either OS WLM or with the new workload management dispatcher in DB2 V10.1.

- When using the DB2 workload management dispatcher, typically soft shares are used to allow higher priority work to utilize any spare capacity available on the system.

Further reading

- Best Practices for multi-temperature data management: http://www.ibm.com/developerworks/data/bestpractices/multitemperature/index.html
- Best Practices for Physical Database Design: http://www.ibm.com/developerworks/data/bestpractices/databasedesign/
- Best Practices for Workload Management: http://www.ibm.com/developerworks/data/bestpractices/workloadmanagement/
- Best Practices for Building a Recovery Strategy for an IBM Smart Analytics System Data Warehouse: http://www.ibm.com/developerworks/data/bestpractices/isasrecovery/index.html
- Best Practices for DB2 for Linux, UNIX, and Windows: http://www.ibm.com/developerworks/data/bestpractices/db2luw/

Contributors

Kevin Beck
Senior Software Engineer, Optim Performance Manager

Paul Bird
Senior Technical Staff Member, Optim & DB2 for Linux, UNIX, and Windows Development

David Kalmuk
Technical Lead / Architect - DB2 for Linux, UNIX, and Windows Process Model, Monitoring, Workload Management

Maksym Petrenko
DB2 Warehouse Integration Specialist

Scott Walkty
DB2 for Linux, UNIX, and Windows Software Developer

Francis Wong
DB2 for Linux, UNIX, and Windows Software Developer

Javier Lozano
Optim Data Studio Developer

Kathy Zeidenstein
Information Development Manager, Data Studio and InfoSphere Warehouse

Notices

Trademarks