IBM® DB2® for Linux, UNIX, and Windows

Best practices:

Expanding an IBM Smart Analytics System database and redistributing data

Austin Clifford
DB2 Data Warehouse QA Specialist

Garrett Fitzsimons
Data Warehouse Specialist

James Cho
Senior Technical Staff Member
IBM Smart Analytics System
IBM Beaverton Lab

Rob Causley
Information Developer
IBM Toronto Lab

Sermsak Sukjirawat
WW Data Warehousing &
IBM Smart Analytics System specialist

Issued: October 2013

Table of Contents

Executive summary

Data warehouse environments continue to experience an explosion in data growth. As a result, you might need additional storage capacity to cope with increased enterprise demands. To help you meet these demands, the IBM® Smart Analytics System is a flexible data warehousing solution that supports a building-block approach to expansion. An important follow-up activity to that expansion is ensuring that you redistribute the data across all database partitions.

To add storage capacity to an IBM Smart Analytics System, you scale out the data warehouse by adding a new data module, across which you expand the existing database. Naturally, you then must redistribute the existing data across the database partitions. This activity reduces the amount of storage that is used on each data module and allows for continued growth of the database across the entire data warehouse.

The objective of adding storage capacity is to accommodate additional data in the database. You might initially experience an increase in performance because each database partition has proportionally less data to process than before the expansion. However, as you continue to add data, performance levels will revert to what they were prior to the expansion. Adding storage capacity does not make the server go faster.

This paper recommends best practices for the process of expanding the database and redistributing data following the hardware build, installation, and cluster expansion[1]. This paper is perhaps more prescriptive than other best practices papers in that it is also an end-to-end guide of the entire redistribution process. Using the approach recommended in this paper for redistributing data entails an outage on the database, so it is essential to get it done as efficiently as possible and to avoid any problems along the way. This paper does not cover manually repartitioning data online in a database.

The key to ensuring a successful database expansion and data redistribution is good planning. This paper covers the prerequisite steps for a successful, timely redistribution operation, such as preparing your database and gathering information. This paper is accompanied by a set of scripts that gather much of the information that you need and provides instructions on how to interpret that data, for example, to estimate how long redistribution will take. Another important factor is tuning the performance of the redistribution operation. By making the temporary configuration changes, such as increasing your sort heap and utility heap sizes and by using the recommended command syntax, you can improve the speed of the operation. By understanding what takes place before, during, and after a database expansion as well as the aspects of data redistribution, you can

[1] For a detailed discussion on identifying, planning and preparing for an expansion of your IBM Smart Analytics System, see the *Expanding an IBM Smart Analytics System* paper (http://www.ibm.com/developerworks/data/bestpractices/expandingsmartanalytics/index.html).

best plan for and implement a database expansion and data redistribution process in your environment.

Introduction

This paper deals with two distinct processes: expanding the DB2 data warehouse database across a new data module and redistributing data across the expanded database.

The first section of the document introduces the concept of database expansion and data redistribution and what you must consider when planning a database expansion. The second section of the paper describes how you can use the `REDISTRIBUTE` command with parameters that help maximize performance as well as providing an overview of the scripts. The third section is a detailed guide to database expansion and data redistribution. The appendixes contain sample output and further information to help ensure a successful process.

Two scripts are available with this paper for download from the developerWorks site. These scripts help in pre-analysis and post-analysis of the database, and you should use them to help prepare and plan for the redistribution process. The scripts and the output from these scripts are referenced in this paper.

Use the command examples in this paper as a guideline. To help avoid problems, always create scripts to execute the commands and test all scripts that you use. Also, use utilities such as `nohup` or `screen` so that a connection failure does not cause a command or script to fail.

Download and use the scripts that are available with this paper on developerWorks to assist in the preparation and planning process.

Expanding a database

In a single-generation environment, each data node has identical hardware resources, is configured identically at the operating system level, and hosts the same number of database partitions.

A multi-generation IBM Smart Analytics System environment exists if different generations of IBM Smart Analytics System modules are combined to support the data warehouse. The number of database partitions per server depends on the specific configuration, typically four or eight. Each database partition on each data node hosts the same quantity of business data. Each database partition is configured with the same DB2 software version, configuration parameter values, and file system structures.

When you expand a database to include a new data module, each database partition on the new data module is added to the existing database. When you add a data module to a data warehouse, you must install and configure the data module by using the same operating system, database software, and file system configuration that the existing modules use. You can then add each new database partition to the database, the result of which is a new entry in the `db2nodes.cfg` file for each new database partition. You can then ex-

tend each database partition group and table space object across the newly expanded database to prepare for redistributing data.

Redistributing data

When you add a database partition to an IBM Smart Analytics System data warehouse, you must redistribute data from existing database partitions to the new database partitions. You must ensure that all database partitions continue to hold an equal quantity of data so that the workload is evenly distributed across all database partitions. You use the data redistribution utility to perform this task, as shown in figure 1.

The recommended use of the data redistribution utility in a data warehouse environment is as a non-logged and non-recoverable operation. This approach minimizes the disk space that is required to complete the operation and helps maximize the performance of the utility.

Redistribute data as a non-logged and non-recoverable operation to minimize the disk space that is required and to help maximize performance of the `REDISTRIBUTE` command.

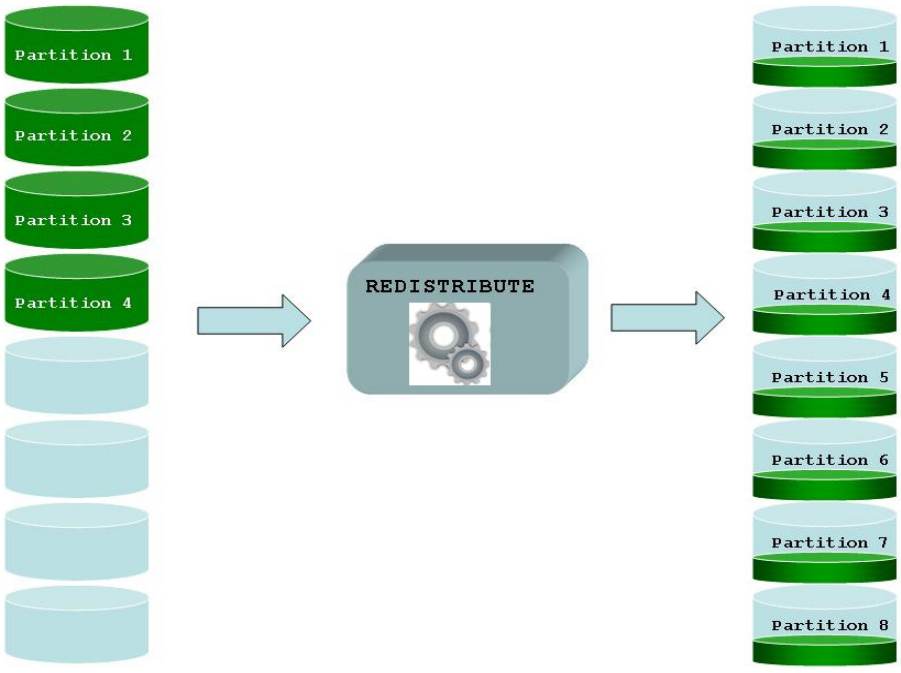

Figure 1. Even redistribution of data from existing to target database partitions

To take advantage of the most recent improvements in REDISTRIBUTE command functionality, upgrade to the currently supported DB2 software release and fix pack for your environment. For details on supported software in your environment, see the "Further

reading" section. The REDISTRIBUTE command on which this paper is focused is available in DB2® for Linux, UNIX, and Windows Version 9.7 Fix Pack 5 and above.

Planning and preparation

During the planning and preparation phase of the project, you must answer the following questions:

- What must change when I expand the database?
- What must happen before I can redistribute data?
- How much disk space do I need?
- What steps are involved in database expansion?
- What steps are involved in data redistribution?
- How long will the process take?

What must change when I expand a database?

An expanded database contains additional database partitions and is positioned for further data growth. Ensure that your existing infrastructure supports the expanded database from a maintenance and applications perspective so that existing service-level objectives for maintenance tasks and ETL performance are still adhered to.

For example, take the following steps:

- Ensure that the existing backup infrastructure accommodates any new database partitions. You might require additional tape drives or storage. For example, if your backup policy is to have a ratio of one tape drive per data node, you need an additional tape drive for each new data node to maintain this ratio.
- Upgrade all scripts that reference the data warehouse to take into account the expanded database, and test these scripts. These scripts should include ETL, backup, recovery, system, database, and maintenance scripts.

Ensure that your backup infrastructure continues to support your recovery time objectives (RTOs), and upgrade and test all relevant scripts after expansion.

What must happen before I can redistribute data?

To help ensure a successful redistribution of data, you must understand the characteristics of your database so you can choose the steps that are most appropriate to your environment.

- Use the precheck script (`redist_precheck.ksh`) that is referenced in this paper to gather data to help determine the approach and detailed steps that you need and to help anticipate any problems. See Appendix C for more information.

- Prune your database where possible; remove unused indexes and tables, partitioned explain tables, and staging tables; and prune the recovery history file.
- Schedule any data archiving process before rather than after data redistribution to reduce the volume of data to redistribute.
- Know which database partition groups you intend to redistribute. In a default IBM Smart Analytics System configuration, you must redistribute data in the database partition groups PDPG and IBMDEFAULTGROUP.
- Understand your acceptance criteria, and prepare and execute any performance query baseline tasks in advance.

Determine which database partition groups to redistribute, and prune the database to ensure that all unnecessary data is removed.

How much disk space do I need?

The amount of available disk space needed depends on the command parameters you choose and the approach you take when redistributing data. Table 2 highlights the characteristics of the different approaches:

Approach	Description: (1) Key characteristics and (2) Typical case
INDEXING MODE DEFERRED	Key characteristics: Only table data is redistributed. Indexes will be marked bad and will need to be rebuilt after the redistribute process has completed. This process of rebuilding indexes could be done during or after the redistribute process or activated on first DML access. Note that where local indexes are implemented, first access only rebuilds the index for the data partition accessed. Typical case: There is not enough disk space remaining to rebuild all indexes in parallel for the table with the largest combined size of indexes. The best approach is to redistribute the table data, reclaim the space and then rebuild the indexes before during or after the database is returned to the business.
Index Mode Rebuild	Key characteristics: Table data is redistributed and all indexes for each table are rebuilt. The data redistribution utility process all indexes for a table in parallel and so is faster than the REORG command which rebuilds each index for a table in sequence. Typical case: I have run the precheck script and have determined that there is enough storage capacity to rebuild all indexes for the largest table in parallel.
Staged approach	Key characteristics: The redistribution of data is performed over multiple outages using either method above. The environment consists of multiple star schemas and tables are logically grouped by shared Distribution Key and included in each redistribution process. Typical case: The elapsed time for the redistribution of all data exceeds the

	time window allowed. To mitigate against this, tables are logically grouped together and redistributed using the `TABLE` parameter of the `REDISTRIBUTE` command. Log archiving would remain on.

Table 1. How your approach affects the amount of storage capacity needed.

This paper focuses on and shows examples of the `INDEXING MODE DEFERRED` approach referenced in table 2 as the approach can be applied in all cases. If you use the `INDEXING MODE DEFERRED` command syntax, additional storage capacity for rebuilding table indexes is not required for the `REDISTRIBUTE` command to complete. Prior to rebuilding indexes post redistribution you might need to reclaim space on the original data nodes.

Use the `INDEX MODE REBUILD` command parameter where sufficient storage capacity exists to rebuild all indexes in parallel for the largest table in your database partition group.

In addition to these requirements, ensure that there is a minimum 100 MB of free space in the `/db2path/bcuaix/NODE`*nnnn* file system for the duration of the redistribution process. This free space is used for generating redistribution control logs that are used for recoverability, and if the redistribution operation runs out of space, it fails.

What steps are involved in database expansion?

The process of expanding the database can take place after you add each new data module to the cluster and validate the data modules. No data is redistributed during this phase. Database expansion can take place as a distinct step before the redistribution process. It is recommended to expand the database a week before the redistribution process to help ensure that all applications and administration tasks operate successfully in the expanded database and that any issues that might arise do not have a material effect on the amount of time needed to redistribute the data. Expanding the database involves the following activities:

1. Adding each new database partition to the database
2. Expanding partitioned database partition groups
3. Extending temporary table spaces
4. Configuring any non-automatic storage table spaces
5. Modifying the backup processes to incorporate each new data module
6. Rebuilding the domain in a high availability (HA) environment

Take the time between database expansion and data redistribution to ensure that all scripts, applications, tasks, and HA and other processes function as expected.

Any new table space created during this database expansion is assigned to the existing partition map and must be processed by the `REDISTRIBUTE` command.

What steps are involved in data redistribution?

When the database has been handed over to the administration function to start the redistribution process, disconnect all query and other application workloads, and then follow a number of distinct steps, as summarized in the following list:

Before the redistribution:

1. Collect performance baseline metrics such as execution times and information such as explain plans for critical queries by using your chosen methods for gathering performance data.
2. Stop all ETL and other scheduled database processes.
3. Drop replicated MQTs.
4. To avoid locking, stop event monitors and other monitoring tools such as Optim Performance Manager (OPM).
5. Run the precheck script, and address any warnings and errors listed in the output.
6. Temporarily modify database configuration settings.
7. Disable HA for the duration of the redistribution process.
8. Create a full online or offline database backup immediately before redistribution. The business can continue to access reports during an online backup operation.

Issue the `REDISTRIBUTE` command as recommended.

After the redistribution:

1. Rebuild indexes, and reduce each table space to the high watermark (HWM).
2. Reset the database configuration parameters to their original values
3. Run the postcheck script (`redist_postcheck.ksh`) and compare its output with the precheck script output, and resolve any errors or warnings.
4. Complete database backup; update statistics; re-create and refresh MQTs.
5. Restart processes, enable high availability and event monitors, and return the database to the business.
6. After you return the database to the business, the process of restarting ETL and then administration processes can take place. At this point, the business should complete a performance baseline test to ensure that query performance after redistribution is equal to or better than baseline query performance before redistribution.

How long will the process take?

The process is divided into two sections: expanding the database and redistributing table data. The amount of time that you require to complete the expansion process and redistribution of data is governed by the following factors:

- Volume of data to redistribute

  The larger the amount of data that each database partition must send to the target database partitions, the longer the operation will take.

- IBM Smart Analytics System model and number of source and target database partitions.

  Newer generations of the IBM Smart Analytics System take advantage of improvements in technology and have a greater capacity to process data.

- Post-redistribution tasks

  Post-redistribution tasks such as rebuilding indexes must take place before you can return the database to the business. Compare the output from the precheck and postcheck scripts to help ensure that the same number and type of objects exist and that the database configuration parameter values are reverted to their original values.

- Elapsed time for backup

  A full database backup is recommended before and after redistribution.

- Characteristics of database to be processed

  The number and ratio of tables, data partitions, MDC tables, and MDC cells in each database partition group affect the total redistribution time.

The precheck script captures data to help estimate elapsed time for database expansion and data redistribution.

The data redistribution utility

In a partitioned database environment, every row is mapped to a database partition by using a hash function. The mapping is described in a partition map. When you issue the `REDISTRIBUTE DATABASE PARTITION GROUP` command a new partition map is created to include each new database partition. Any table spaces created after the new partition map is created use the new partition map. To complete the expansion of the data warehouse, you must redistribute the data in the original tables to adhere to the new partition map.

To avoid locking, redistribute data during a maintenance window; planned downtime is preferable to unplanned downtime.

You should perform redistribution during a maintenance window. Parallel querying of data might cause both the redistribution and queries to slow down significantly. Redistributing the data online can also cause lock timeouts or deadlocks when maintenance or complex query operations are in progress.

The `REDISTRIBUTE` command acquires an exclusive lock (z-lock) on each table to begin the redistribution of data. If a lock cannot be acquired, the command fails. In this situation, you must address the cause of the problem (for example, an event monitor) and restart the `REDISTRIBUTE` command. Processing then restarts from the beginning of the table on which the operation failed. When processing is completed for a table, all affected table spaces are placed in backup pending state.

Using a single `REDISTRIBUTE DATABASE PARTITION GROUP` command, it is possible to redistribute the data in all or a subset of tables in a database partition group. The data redistribution utility has the following features:

- It automatically creates new objects and mapping dependencies such as foreign keys, partitioned MQTs, and triggers, and grants privileges.
- Data is automatically reorganized on each node. Free space that is generated as a result of relocating records is filled immediately with records from the logical end of the table.
- You can restart a redistribution operation. You can interrupt the process by pressing Ctrl+C and then resume the redistribution by specifying the `REDISTRIBUTE DATABASE PARTITION GROUP` command with the `CONTINUE` parameter.

Recommended REDISTRIBUTE command syntax

Use the following command syntax to redistribute user data in a database partition group:

```
REDISTRIBUTE DATABASE PARTITION GROUP db-partition-group
NOT ROLLFORWARD RECOVERABLE INDEXING MODE DEFERRED DATA
BUFFER 180000 STATISTICS NONE QUIESCE DATABASE YES
```

Use the recommended `REDISTRIBUTE` command syntax to help ensure that you successfully redistribute data every time.

The following list provides more information about the parameters in the recommended command syntax:

- `NOT ROLLFORWARD RECOVERABLE`

  The recommended redistribution method is not rollforward recoverable to avoid the extensive logging that results from a log record being created for every relocated table row. If a hardware failure occurs and the database is lost or corrupted, you must restore from the backup that you took before redistribution started.

- `INDEXING MODE DEFERRED`

  Most data warehouse environments are characterized by a single or small number of fact tables that represent a large proportion of the overall database size. Rebuilding the indexes for each large fact table in parallel requires a significant amount of disk space, which is not always available. If storage capacity is an issue, it is preferable to rebuild all indexes after redistribution.

  Use the `INDEX MODE REBUILD` parameter if you wish to rebuild indexes as part of the redistribution process.

- `DATA BUFFER`

  Specify a `DATA BUFFER` parameter value of at least 180,000 pages, up to 90% of the recommended utility heap size.

- `STATISTICS NONE`

  In a warehouse environment, it is more efficient to perform the `RUNSTATS` statistics collection after redistribution.

- `QUIESCE DATABASE YES`

  Specifies to force all users off the database and put it into a quiesced mode.

Test scenarios

You should test all commands, scripts, and steps in a non-production system and familiarize yourself with the process in your environment.

Complete a test run in a non-production environment to validate the steps, commands, scripts, and process that you intend to use in production.

When testing data redistribution on a non-production system, consider using the following approach:

- Include those tables that are identified by warning messages in the precheck script output.
- Include your largest fact and dimension tables and tables that combine the MDC and range-partitioning features. These tables are listed in the precheck script output.
- Use the `TABLE` parameter of the `REDISTRIBUTE` command, as shown in the following example, to redistribute table data for just those tables that you want to include in the test:

```
REDISTRIBUTE DATABASE PARTITION GROUP db-partition-group NOT
ROLLFORWARD RECOVERABLE UNIFORM INDEXING MODE DEFERRED TABLE
("schema"."table") ONLY DATA BUFFER 180000 STATISTICS NONE
```

- Determine how much time it takes for your largest and representative average replicated MQTs to be refreshed.
- Determine how much time it takes for statistics (`runstats`) collection to complete on your largest fact and dimension tables.
- Determine how much time it takes to complete an index rebuild on your largest fact and dimension tables.

Redistribution-related scripts

Use the precheck script to help you prepare to expand your database and redistribute data. Use the postcheck script to review the database status after redistribution and to compare database characteristics before and after redistribution.

Obtain and set up the scripts as follows:

1. On the administration node, create a work directory for the scripts and output files by issuing the following command as DB2 instance owner:

```
mkdir $DB2HOME/IBMredist
```

2. Download the precheck and postcheck scripts into the `/IBMredist` directory.
3. Make the scripts executable, as shown in the following example:

```
chmod +x redist_precheck.ksh
chmod +x redist_postcheck.ksh
```

The precheck script

You can run the precheck script as often as needed to monitor progress during the redistribution project. This script provides an estimate of the length of time that the `REDISTRIBUTE` command takes to run. By default, the script accesses the system catalog tables on the coordinator node only. Statistics stored in the system catalog tables are used to determine the state of your database. It is important that all statistics are current to help ensure that the data gathered by the precheck script is accurate.

The accuracy of the precheck output is dependent on statistics for all tables in the database partition group being current.

You must run only a single instance of the precheck script at any one time to avoid locking any system catalog tables.

Run the precheck script for each database partition group that you intend to redistribute. The following example generates output for the database partition group `PDPG` on the database `BCUDB`, where eight database partitions are to be added:

```
./redist_precheck.ksh BCUDB PDPG 8
```

Each execution of the precheck script generates a unique output file in the directory in which you ran the script. An example of a file name is `redist_precheck_bcudb_pdpg_20120506045612.out`. A copy of the output on which this paper is based is in Appendix E.

The output file contains informational, warning, and error messages. The informational messages provide information about your database related to factors that influence how the redistribution operation will perform. The warning messages indicate conditions that might affect the performance of the `REDISTRIBUTE` command; you should review these.

Error messages are reported for conditions that would cause the `REDISTRIBUTE` command to fail; you must address these messages before issuing the `REDISTRIBUTE` command.

An optional parameter, `FULL`, checks the state of all table spaces in the database partition group. If you specify this parameter, the script takes longer to run because it checks the state of each table space on each database partition. You should use this parameter directly before you issue the `REDISTRIBUTE` command.

Use the `FULL` parameter of the precheck script, review all warning messages, and address all error messages immediately before you issue the `REDISTRIBUTE` command.

You should also generate precheck output for the IBMDEFAULTGROUP database partition group. The contents of that database partition group should be negligible. If it is not, perhaps a table space was created in error.

Informational messages for the precheck script

The purpose of the informational messages in the precheck output file is to help you understand the characteristics of the database that you will expand and redistribute. In particular, you should review any messages related to the following topics:

- **DB2 software level, fix pack, and special build, if applied**

  The existing software level is shown. Ensure that you upgrade to the latest validated version for your stack or to the fix pack that is recommended by IBM Support to take advantage of all the latest enhancements to the redistribute command.

- **Average elapsed time in the current year for a full database backup on a database partition**

  Use this information to determine how much time is needed to back up the database before and after the redistribution process.

  If you do not reclaim space on the original data nodes before you take the post-redistribution backup, the backup time might not decrease for the original database partitions. You must first reduce each table space to the new (lower) HWM.

- **Top ten tables by size**

  Information about the characteristics of the largest tables is provided. These tables take the most time to redistribute. If these tables have many range partitions or are range partitioned and use MDC functionality, data redistribution can take up to two times for these tables than for standard tables.

- **Top ten tables by `fpages` size**

  MDC tables that have an `fpages` to *`npages`* ratio of 2:1 are deemed to be sparsely populated. Creating and redistributing data for these sparsely populated MDC tables can increase the elapsed time needed to complete the redistribution

process. Refer to the `PAGE_RATIO` column in this area of the precheck output to identify tables in this category.

- **Statistics**

  Ensure that your statistics are as up to date as possible. Updated statistics help determine how accurate the row count statistics are in the precheck script output; the more outdated the statistics are, the less reliable the precheck estimates and recommendations are likely to be. The precheck script uses the system catalog tables only, and these statistics depend on the last execution of the `RUNSTATS` command for each table.

- **Data skew at database partition and database partition group level**

  Data redistribution does not correct data skew. If you implement a custom partition map to manipulate data placement, you should perform a detailed test to ensure that the redistribution utility performs as required. Because you are expanding your database, any data skew on your database before redistribution might be exaggerated after redistribution.

  If data skew exists within a database partition group at the database partition level, ensure that it does not exceed 10%, otherwise you should investigate and correct any faults before you perform data redistribution. For more information about identifying data skew, see the "Further reading" section.

Data redistribution does not correct data skew. Address any data skew issues that are caused by distribution key selection or implementation of a custom partition map before proceeding with data redistribution.

- **Volume of data in the specified database partition group**

  The precheck script outputs the amount of table data that you must redistribute. Index data is not included. The script also provides an overview of table spaces and the overall size of the database. This information might highlight some outstanding maintenance tasks that you could perform before data redistribution.

- **Enumeration of tables by type**

  Each table on each database partition, however small the table, requires a small amount of maintenance time to be created on each new database partition. This time can be up to 5 seconds per table. If a large number of tables exist or a large number of range partitions exist, you should be aware of the effect that this has on the overall redistribution time. A large number of range partitions can exist if you create a range partition for week or day.

- ***N* minutes are estimated for the redistribution of this database partition group**

  The precheck script uses the formula described in Appendix A to estimate the number of minutes for the `REDISTRIBUTE` command to run.

Warning messages for the precheck script

The precheck script can generate the following warning messages. Address the warnings that are reported in the output file by using the following guidelines. To verify that you resolved each issue that was reported, rerun the precheck script.

- **The `logarchmeth1` configuration parameter is not set to OFF. Set it to OFF for redistribution.**

  You should turn off log archiving before running the redistribute command. This setting enables indexes to be rebuilt before the post-redistribution backup.

- **The `util_heap_sz` configuration parameter setting is low. Consider increasing it to `200000` before redistribution.**

  You should tune the utility heap before running the redistribute command.

- **The `sortheap` configuration parameter setting is low. Consider increasing it to 20000 for redistribution.**

  You should tune the sort heap size before running the redistribute command.

Error messages for the precheck script

The precheck script can generate the following error messages. Each error will cause the `REDISTRIBUTION` command to fail. Address the errors that are reported in the output file by using the following guidelines. To verify that you resolved each issue that was reported, rerun the precheck script.

- **Tables are in abnormal state, check pending state, or pending state.**

  The redistribute command fails if it encounters a table in an abnormal state. Before issuing the redistribution command, ensure that all tables are in a normal state.

- **Replicated MQTs exist. Use the db2look command to generate the DDL statements for replicated MQTs, and drop the MQTs before redistribution.**

  Replicated MQTs are not supported in a redistribution operation, so you must drop replicated tables that are not partitioned before issuing the redistribute command. Re-create the tables on all database partitions after the redistribute command has run.

- **Event monitors are running. Stop them before redistribution.**

  Disable all event monitors and monitoring processes such as OPM before the redistribution process. Locking might occur if these processes are enabled.

- **DMS table spaces do not have the auto-resize feature enabled.**

  The redistribute command resizes table spaces where required. If you do not set the AUTORESIZE parameter for a DMS table space, as shown in the following example, the redistribute command fails.

  ```
  ALTER TABLESPACE TS AUTORESIZE YES
  ```

- **Table spaces in abnormal state**

The redistribute command fails if it encounters a table space in an abnormal state. Before issuing the redistribution command, ensure that all table spaces are in a normal state.

The postcheck script

You can use the postcheck script to perform a before-and-after comparison of the database. You can run the postcheck script multiple times following the completion of the `REDISTRIBUTE` command. The script does not change the environment on which you run it.

Similar to the execution of the precheck script, each execution of the postcheck script generates a unique output file with informational messages, warning messages, and error messages. The conditions that trigger warning messages might affect database behavior, so you should address them. You must address error messages before returning the database to production use.

Run the postcheck script for the database and database partition groups that you intend to redistribute. The following example generates output for the database partition group PDPG on the database BCUDB:

```
./redist_postcheck.ksh BCUDB PDPG
```

An output file, for example, `redist_postcheck_bcudb_pdpg_20120306045612.out`, is created in the directory in which you ran the script.

Informational messages for the postcheck script

Compare the informational messages in the postcheck output file to the informational messages in the precheck output file to ensure that all database objects are in the same state before and after redistribution.

Warning messages for the postcheck script

The postcheck script can generate the following warning messages. Address the warnings that are reported in the output file by using the following guidelines. To verify you resolved each issue that was reported, rerun the postcheck script

- **Database partition groups are in the redistribute state.**

  The database partition group should not be in a redistribute in progress state after the redistribute command has finished running.

- **The logarchmeth1 configuration parameter is set to OFF. Change the value to the pre-redistribution value.**

  Revert the setting for logarchmeth1 to its pre-redistribution value.

- **The logarchmeth2 configuration parameter is set to OFF. Change the value to the pre-redistribution value.**

  Revert the setting for logarchmeth2 to its pre-redistribution value.

- **The util_heap_sz configuration parameter setting is high. Check the setting before returning the system to production.**

  You should change the utility heap size to the pre-redistribution value.

- **The sortheap configuration parameter setting is high. Check the setting before returning the system to production.**

  You should change the sort heap size to the pre-redistribution value.

Detailed guide to expanding a database and redistributing data

The examples provided in this section are used to add a data node with eight database partitions to an existing IBM Smart Analytics System. The database partitions are numbered 17 - 24. The default instance name and database name for the IBM Smart Analytics System 7700, bcuaix and BCUDB, are also used.

To help ensure that you prepare as recommended in this paper, use the precheck script at regular intervals before issuing the `REDISTRIBUTE` command to monitor your progress and identify if any characteristics of the database have changed.

The steps in this section are based on the assumption of a default configuration of an IBM Smart Analytics System. The steps are as follows:

1. Add each database partition on the new data node to the database.
2. Expand the database objects to include the new database partitions.
3. Finish preparing the environment for data redistribution.
4. Issue the `REDISTRIBUTE` command.
5. Perform post-redistribution steps.
6. Return the database to the business.

Adding a data node and expanding the database

Complete this process after racking, installing, and making the hardware and software for each new data node available on your cluster. An outage is not necessary, but you should perform these tasks during an off-peak period and recycle the instance as a final step to ensure that there are no issues.

This process does not require a significant amount of time: typically a minute or two to add each database partition to each new data node.

When you add a database partition, the database partition is activated. The time that is required to activate the database partition depends on the size of the logs because the logs must be allocated. If you want to add many database partitions and have large log files (more than 1 GB per database partition), consider temporarily reducing the value of the `logfilsiz` parameter, `logprimary` parameter, or both. For example, the following commands could be used:

```
UPDATE DB CFG FOR BCUDB USING LOGFILSIZ 1000
UPDATE DB CFG FOR BCUDB USING LOGPRIMARY 10
ACTIVATE DATABASE BCUDB
```

Expand the database separately from redistributing data. This separation allows time to address any expansion-related issues without affecting the timeline for data redistribution.

You should expand the database after you add the new hardware to the cluster. The process of expanding the database can be seen as validation of the expansion process. You should expand the database separately from redistributing the data so that you can validate all modified scripts, applications, failover functionality, and processes before redistributing data.

Back up the instance configuration before making any database changes. For more details, see the "Further reading" section.

Step 1. Add each database partition on the new data node to the database

To add each database partition to the database:

Issue the `db2start DBPARTITIONNUM` command for each partition. As illustrated in the following example, which adds the data node `BCUDATA03`, add the database partition with the lowest number first, and add the other database partitions in ascending order of partition number:

```
db2start DBPARTITIONNUM 17 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 0 WITHOUT TABLESPACES

db2start DBPARTITIONNUM 18 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 1 WITHOUT TABLESPACES

db2start DBPARTITIONNUM 19 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 2 WITHOUT TABLESPACES

db2start DBPARTITIONNUM 20 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 3 WITHOUT TABLESPACES

db2start DBPARTITIONNUM 21 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 4 WITHOUT TABLESPACES

db2start DBPARTITIONNUM 22 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 5 WITHOUT TABLESPACES

db2start DBPARTITIONNUM 23 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 6 WITHOUT TABLESPACES

db2start DBPARTITIONNUM 24 ADD DBPARTITIONNUM HOSTNAME
BCDATA03 PORT 7 WITHOUT TABLESPACES
```

Use the following guidelines to determine the specifications for your command:

- `HOSTNAME` refers to the host name of the data node to add. Use the host name that is associated with the fast communication manager (FCM) network, not the management or corporate network. HOSTNAME is case sensitive in TSA and therefore the case used in the commands above should match that in TSA.
- `PORT`, followed by a number, refers to the port to use for an individual database partition: `0` for the first database partition on the new data node, `1` for second database partition on the new data node, and so on until all 8 database partitions on the new node are added.
- The `WITHOUT TABLESPACES` clause. You must specify this clause for an IBM Smart Analytics System because the administration node can contain system temporary table space containers that are different from those on the data nodes. If you do not specify this clause, an error results.

Each new database partition inherits the database configuration settings of the database.

The process of adding entries to the `db2nodes.cfg` file is completed automatically as part of this process. Manually adding entries to the `db2nodes.cfg` file is not supported in a partitioned database environment.

If you reduced the value of the `logfilsiz` or `logprimary` parameters, change the values back to the original setting.

If a database partition does not start successfully when you issue the `db2start` command, diagnose and address any reported issues

Step 2. Recycle the instance

You must recycle the instance so that the new database partitions can participate in the partitioned database environment.

You should see each new database partition listed in the stop and start database processes.

1. Ensure that the database is activated.
2. Confirm that the new database partitions have been added to the instance by checking the `~/sqllib/db2nodes.cfg` file.

Step 3. Alter the TEMP16K table space

In an IBM Smart Analytics System or PureData System for Operational Analytics , you must manually extend the default DMS temporary table space, TEMP16K, across each new database partition.

To alter the TEMP16K table space in a IBM Smart Analytics System, adapt the following example:

```
ALTER TABLESPACE TEMP16K ADD (FILE '/db2fs/bcuaix/NODE00 $N
/BCUDB/temp16k' 5G) ON DBPARTITIONNUMS (17 to 24) AUTORESIZE
YES
```

where:

- `'/db2fs/bcuaix/NODE00 $N /BCUDB/temp16k'` represents the file system and path that are associated with the database.
- `DBPARTITIONNUMS (17 to 24)` denotes the range of database partitions that were added to the database.

To alter the TEMP16K table space in a PureData System for Operational Analytics, which uses a combination of SSD and external storage, adapt the following example:

```
ALTER TABLESPACE TEMP16K ADD (FILE '/db2ssd/bcuaix/ssd $N%8
/BCUDB/temp16k' 5G) ON DBPARTITIONNUMS (17 to 24) AUTORESIZE
YES

ALTER TABLESPACE TEMP16K BEGIN NEW STRIPE SET (FILE
'/db2fs/bcuaix/NODE00 $N /BCUDB/temp16k' 10G) ON DBPARTI-
TIONNUMS (17 to 24)
```

where:

- `'/db2ssd/bcuaix/ssd $N%8 /BCUDB/temp16k'` represents the file system and path for the containers on SSD storage that are associated with the database.
- `'/db2fs/bcuaix/NODE00 $N /BCUDB/temp16k'` represents the file system and path for the external storage that are associated with the database.

Step 4. Expand the PDPG and IBMDEFAULTGROUP database partition groups

Expand the PDPG and IBMDEFAULTGROUP database partition groups in sequence across all the database partitions. For example:

1. Issue the ALTER DATABASE PARTITION GROUP statement for the PDPG database partition group. An example follows.

```
ALTER DATABASE PARTITION GROUP PDPG ADD DBPARTITIONNUMS (17
TO 24) WITHOUT TABLESPACES
```

2. Issue the ALTER DATABASE PARTITION GROUP statement for the default DB2 database partition group. An example follows:

```
ALTER DATABASE PARTITION GROUP IBMDEFAULTGROUP ADD DBPARTI-
TIONNUMS (17 TO 24) WITHOUT TABLESPACES
```

This statement generates warning message `SQL1759W`. In this context, this message is informational; you can ignore it.

Step 5. If applicable, expand any non-automatic storage table spaces

If your database contains DMS table spaces, you must alter those table spaces.

To alter the table spaces, use the following statement as an example:

```
ALTER TABLESPACE <ts_name> ADD (FILE '/db2fs/bcuaix/NODE00
$N /BCUDB/<ts_name_ts>' 5G) ON DBPARTITIONNUMS (17 to 24)
```

This statement generates warning message `SQL1759W`. In this context, this message is informational; you can ignore it.

Step 6. If applicable, rebuild the HA domain

If HA is implemented, you need to rebuild the domain to ensure that all database partitions are recognized.

To complete this task, see your IBM Tivoli System Automation for Multiplatforms (SA MP) documentation, and work with IBM Support and Lab Services. As an output of this task, you should have an updated cluster architecture installation record and a complete HA test cases document.

Finish preparing the environment for redistribution

The time available between database expansion and data redistribution should be used to test backup and other maintenance and application processes to help ensure that any changes made to accommodate the additional data nodes and database partitions complete as expected.

You must perform the following steps immediately before issuing the redistribution command.

Step 1. Capture database DDL statements

To capture the information required to re-create any replicated MQTs, use the following procedure:

1. Capture the DDL statements for the warehouse by issuing the `db2look` command, as shown in the following example:

   ```
   db2look -d dbname -e -o redist_db2look.out
   ```

   where `redist_db2look.out` is the name of the output file.

2. Extract the DDL statements for the replicated MQTs from the output file. You will use these DDL statements later, after data redistribution completes, to re-create the replicated MQTs.

3. Create a list of all replicated MQTs that must be dropped, as shown in the following example:

   ```
   SELECT RTRIM(TABSCHEMA)||'.'||TABNAME FROM SYSCAT.TABLES
   WHERE PARTITION_MODE='R'
   ```

4. Generate the DDL to re-create each replicated MQT, as shown in the following example:

```
db2look -e -d BCUDB -x -z <SCHEMA> -tw <TABLE> -o
<SCHEMA>.<TABLE>.txt
```

Step 2. Back up the database

You can take an online or offline backup.

If a hardware or similar failure occurs and you must restore the database, a simple and timely recovery process that returns the database to the production state immediately before the redistribution step is desirable. Minimizing the number of logs to be replayed is an important part of achieving this objective.

To back up the database:

1. If you plan to perform an online backup, stop all ETL processes, although the database can be available to the business in other ways during the backup operation.
2. Connect to the database.
3. If you plan to do an offline backup, quiesce the database:

```
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS
```

4. If you plan to do an offline backup, verify that no applications are connected to the database by issuing the following command and checking that there is no output:

```
LIST APPLICATIONS SHOW DETAIL
```

5. Back up the database by using your standard method. The following example shows a backup command that backs up all of the database partitions and specifies Tivoli Storage Manager as the target device:

```
BACKUP DATABASE BCUDB ON ALL DBPARTITIONNUMS ONLINE USE TSM
INCLUDE LOGS
```

Step 3. Stop active processes

To stop active processes:

1. Disable any event monitors or active connections. For example, to disable the deadlock event monitor, issue the following statement:

```
SET EVENT MONITOR DB2DETAILDEADLOCK STATE 0
```

2. Stop Optim Performance Manager (OPM) by using the following command:

```
/opmfs/IBM/OPM/OPMstop.sh
```

3. To prevent statistics gathering from locking a table that you will redistribute, set the `auto_runstats` configuration parameter to `OFF`:

```
UPDATE DB CFG FOR BCUDB USING auto_runstats OFF
```

Step 4. Allocate additional memory to the REDISTRIBUTE command

To allocate additional memory to the `REDISTRIBUTE` command for the duration of the redistribution process:

1. Identify database configuration parameters whose values are outside the recommended values by using the precheck script output.
2. Temporarily tune the values of the indicated database configuration parameters.

Increase the utility heap and sort heap memory allocations for the duration of the redistribution process.

The following settings can help increase performance:

- **Utility heap**

  The utility heap determines the amount of memory that is available for utilities. Set the utility heap to at least 200,000 pages, as illustrated in the following sample command:

  ```
  UPDATE DATABASE CONFIGURATION FOR BCUDB USING UTIL_HEAP_SZ
  200000
  ```

  The value of the util_heap_sz configuration parameter value should be at least 10% larger than the value of the `REDISTRIBUTE` command **`DATA BUFFER`** parameter.

- **Sort heap**

  Sort heap memory is used when indexes are rebuilt. Set the `sortheap` database configuration parameter to a value of `10000` – `20000`, depending on the number and size of indexes for your largest tables.

  ```
  UPDATE DATABASE CONFIGURATION FOR BCUDB USING SORTHEAP
  20000
  ```

- **Log archiving**

  Disable log archiving for the duration of the redistribution process to allow indexes to be rebuilt before you complete the post-redistribution backup operation. Examples of how to disable log archiving are as follows:

  ```
  UPDATE DATABASE CONFIGURATION FOR BCUDB USING LOGARCHMETH1
  OFF
  UPDATE DATABASE CONFIGURATION FOR BCUDB USING LOGARCHMETH2
  OFF
  ```

Deactivate and Activate the database to effect the changes to the settings:

```
DEACTIVATE DATABASE BCUDB
ACTIVATE DATABASE BCUDB
```

The postcheck script issues a warning message to remind you to reset the parameters to their original values.

Step 5. Disable HA for the duration of the redistribution process

To avoid an unwanted automatic failover event during the redistribution process, disable HA. The database configuration parameters that are recommended in this paper are designed to assign a large amount of memory to the `REDISTRIBUTE` command, and this can trigger a failover.

Disable HA for the duration of the data redistribution process to avoid an automated failover event that can trigger a failure.

To disable HA:

1. Put Tivoli SA MP into manual mode. On the administration node, as root user, issue the following command:

```
samctrl -M T
```

2. Change the critical resource protection method so that RSCT does not reboot a node if a communication issue affects the cluster. On the administration node, as root user, issue the following command:

```
chrsrc -c IBM.PeerNode CritRsrcProtMethod=1
```

Step 6. Optional: Rerun the precheck script

To check the state of each individual table and table space on each individual database partition, run the precheck script with the `FULL` parameter. This helps avoid any issues that might cause the `REDISTRIBUTE` command to fail. Running a full check takes longer, and using it is recommended immediately prior to issuing the REDISTRIBUTE command or when there is no activity on the system. An example follows:

```
./redist_precheck.ksh BCUDB PDPG 8 FULL
```

When using the FULL option look for an error in the precheck output file that lists a table or table space in an abnormal state.

Issuing the REDISTRIBUTE command

When you use the REDISTRIBUTE command, you redistribute the data across the expanded database by using a new partition map.

To redistribute data:

1. Connect to the database.
2. Issue the following `REDISTRIBUTE` command for the database partition group:

```
REDISTRIBUTE DATABASE PARTITION GROUP db-partition-group NOT
ROLLFORWARD RECOVERABLE UNIFORM DATA BUFFER 180000 INDEXING
MODE DEFERRED STATISTICS NONE QUIESCE DATABASE YES
```

If you are redistributing a database partition group and storage capacity is not a concern, issue the following command, which includes the `INDEX MODE REBUILD` parameter:

```
REDISTRIBUTE DATABASE PARTITION GROUP db-partition-group NOT
ROLLFORWARD RECOVERABLE UNIFORM DATA BUFFER 180000 INDEXING
MODE REBUILD STATISTICS NONE QUIESCE DATABASE YES
```

3. If the REDISTRIBUTE command fails, use the following command as a sample to continue the redistribution process:

```
REDISTRIBUTE DATABASE PARTITION GROUP db-partition-group NOT
ROLLFORWARD RECOVERABLE CONTINUE INDEXING MODE DEFERRED
```

You can monitor progress in two ways:

- Through the `LIST UTILITIES` command
- By looking at the output from the `REDISTRIBUTE` command ,which is located in the `~/sqllib/redist` directory

For more details on monitoring, see the section "Monitoring and troubleshooting data redistribution."

Performing post-redistribution tasks

After the redistribute command has successfully run, you must perform a number of tasks before returning the database to the business.

Step 1. Reclaim MDC extents

To reclaim MDC extents for each table in the database partition group, use the following command as an example:

```
REORG TABLE SCHEMA.TABLE RECLAIM EXTENTS ONLY
```

This command does not perform full table reorganization.

Step 2. Reduce data table spaces to HWM

Reduce the HWM for each table space that contains table data only, indexes are processed in step 4, in each database partition group that you redistributed. Reducing the HWM helps align the number of pages per table space across all database partitions and also helps reduce the elapsed backup operation time on the original database partitions.

Reduce the HWM to align the number of pages that are used by each table space across all database partitions and reduce the elapsed backup time on the original database partitions.

To reduce the HWM:

1. Identify a list of table spaces by type that contain table data in partition group PDPG:

```
SELECT B.TBSPACE, B.TBSPACETYPE FROM SYSCAT.DATAPARTITIONS
A, SYSCAT.TABLESPACES B, SYSCAT.TABLES C WHERE
A.TBSPACEID=B.TBSPACEID AND A.TABSCHEMA=C.TABSCHEMA AND
A.TABNAME=C.TABNAME AND A.DATAPARTITIONID=0 AND B.DBPGNAME=
'PDPG' ORDER BY TBSPACETYPE, TBSPACE
```

2. Issue the following statement for each automatic storage table space:

```
ALTER TABLESPACE TS_NAME REDUCE MAX
```

3. For DMS (non-automatic storage) table spaces, issue the following statements:

```
ALTER TABLESPACE TS_NAME LOWER HIGH WATER MARK
ALTER TABLESPACE TS_NAME REDUCE (ALL CONTAINERS 4G)
```

where 4G represents the amount of free space to reclaim. Issue the following statement to determine the amount of space to reclaim for each DMS table space:

```
SELECT MIN(TBSP_FREE_SIZE_KB) FROM SYSIB-
MADM.TBSP_UTILIZATION WHERE TBSP_NAME = 'TS_NAME'
```

4. Monitor progress. The `ALTER TABLESPACE REDUCE` statement is asynchronous. You can use the `MON_GET_EXTENT_MOVEMENT_STATUS` or `MON_GET_TABLESPACE` monitoring function to track progress:

```
SELECT TBSP_NAME, MEMBER, NUM_EXTENTS_MOVED,
NUM_EXTENTS_LEFT FROM TA-
BLE(SYSPROC.MON_GET_EXTENT_MOVEMENT_STATUS('', -1))
```

Step 3. Rebuild indexes on all tables

In this step, it is assumed that you used the recommended `REDISTRIBUTE` command syntax and that all table spaces are in a normal state.

When rebuilding indexes for multiple tables in parallel, choose tables in different table spaces to avoid index fragmentation within a table space container.

To rebuild indexes across all database partitions in the redistributed database partition group:

1. Issue the following statement:

```
SELECT DISTINCT T.TBSP_NAME, TBSP_STATE FROM TA-
BLE(MON_GET_TABLESPACE('',-2)) AS T WHERE T.TBSP_NAME IN (
SELECT TBSPACE FROM SYSCAT.TABLESPACES WHERE DBPGNAME =
'PDPG') WITH UR
```

2. Issue the `REORG INDEXES` command for each table that has an index, using the following example as a guideline. Process tables by size, starting with the largest table.

```
REORG INDEXES ALL FOR TABLE SCHEMA.TABLE ON ALL DBPARTI-
TIONNUMS
```

If multiple indexes per table exist, they are processed in sequence; this reduces the amount of storage that is required to process indexes for a table.

Step 4. Reduce the HWM for index and MQT table spaces

Reducing the high water mark for table spaces that contain index or MQT data allows you to reclaim free space made available post redistribution. Note that the procedures described below assume DB2 9.7 or higher. If you have DB2 9.5 table spaces, the procedure is more cumbersome and lengthy (see the "Further reading" section). To reduce the HWM for each table space in each database partition group that you redistributed:

1. Identify the list of table spaces in partition group PDPG:

   For MQT tables:

```
SELECT B.TBSPACE, B.TBSPACETYPE FROM SYSCAT.DATAPARTITIONS
A, SYSCAT.TABLESPACES B, SYSCAT.TABLES C WHERE C.TYPE = 'S'
AND C.PARTITION_MODE = 'R' AND A.TBSPACEID=B.TBSPACEID AND
A.TABSCHEMA=C.TABSCHEMA AND A.TABNAME=C.TABNAME AND
A.DATAPARTITIONID=0 AND B.DBPGNAME= 'PDPG'
```

For indexes:

```
SELECT B.TBSPACE, B.TBSPACETYPE FROM SYSCAT.TABLES A,
SYSCAT.INDEXES I, SYSCAT.TABLESPACES B,
SYSCAT.DATAPARTITIONS D WHERE A.TABSCHEMA = D.TABSCHEMA AND
A.TABNAME = D.TABNAME AND A.TABSCHEMA = I.TABSCHEMA AND
A.TABNAME = I.TABNAME AND D.TBSPACEID = B.TBSPACEID AND
DATAPARTITIONID = 0 AND B.DBPGNAME = 'PDPG' WITH UR
```

2. For each automatic storage table space, issue the following statement:

```
ALTER TABLESPACE TS_NAME REDUCE MAX
```

3. For DMS (non-automatic storage) table spaces, issue the following statements:

```
ALTER TABLESPACE TS_NAME LOWER HIGH WATER MARK
ALTER TABLESPACE TS_NAME REDUCE (ALL CONTAINERS 4G)
```

   where 4G represents the amount of free space to reclaim. Issue the following statement to determine the amount of space to reclaim for each DMS table space:

```
SELECT MIN(TBSP_FREE_SIZE_KB) FROM SYSIB-
MADM.TBSP_UTILIZATION WHERE TBSP_NAME = 'TS_NAME'
```

Step 5. Re-create and refresh replicated MQTs

To help generate the commands to re-create replicated MQTs, use the output from the `db2look` command that you issued earlier. The time that it takes to refresh an MQT depends on a number of factors:

- The number of rows in the table
- The network capacity of the administration node to send these rows to each database partition
- The number of database partitions on which the table must be refreshed

If needed, a workaround is to create the base table from which the refresh takes place as a partitioned table before creating the replicated MQT table.

Step 6. Reset database configuration values

To return the database configuration to its state before the redistribution process:

1. Reset the database configuration parameters to the original values, as recorded through your database backup or in the precheck script output. Use the following sample commands as a guideline:

```
UPDATE DATABASE CONFIGURATION FOR BCUDB USING LOGARCHMETH1
TSM
UPDATE DATABASE CONFIGURATION FOR BCUDB USING UTIL_HEAP_SZ
65536
UPDATE DATABASE CONFIGURATION FOR BCUDB USING SORTHEAP
35000
```

```
UPDATE DATABASE CONFIGURATION FOR BCUDB USING AUTO_RUNSTATS
ON
```

2. Switch on event monitors:

```
SET EVENT MONITOR DB2DETAILDEADLOCK STATE 1
```

Returning the database to the business

To return the database to the business, complete the following steps.

Step 1. Back up the database

After redistribution, a full database backup is necessary to have a recoverable database that contains the new database topology.

Back up the database by using your standard method, such as the one in the following example:

```
BACKUP DATABASE database ON ALL DBPARTITIONNUMS TO TSM
```

1. Issue the `RUNSTATS` command against each table, as per your regular statistics policy. An example for an individual table follows:

   ```
   RUNSTATS ON TABLE SCHEMA.TABLE AND INDEXES ALL TABLESAMPLE
   SYSTEM(1) SET PROFILE
   ```

   For FAQs on building a `RUNSTATS` command statistics-gathering strategy, see the "Best practices" link in the "Further reading" section.

Step 2. Flush the package cache

To flush the package cache:

1. Issue the following statement:

   ```
   FLUSH PACKAGE CACHE DYNAMIC
   ```

This statement causes all cached dynamic SQL statements to be marked as invalid, which forces the next request for the same SQL statements to be recompiled

2. Rebind all of the packages in the database using the following command

   ```
   db2rbind database -l logfile all
   ```

Note that a warning message, `SQL0204N`, will be output where the object referenced no longer exits.

Step 3. Re-enable HA

To enable HA:

1. Take Tivoli SA MP out of manual mode. As root user on the administration node, issue the following command:

   ```
   samctrl -M F
   ```

2. Revert to the critical resource protection method. As root user on the administration node, issue the following command:

   ```
   chrsrc -c IBM.PeerNode CritRsrcProtMethod=0
   ```

Step 4. Restart event monitors and applications

To restart any event monitors, use the following command:

```
SET EVENT MONITOR event-monitor-name STATE 1
```

Enable OPM by using the `OPMstart.sh` script on the management node of the cluster.

Configure OPM to collect performance metrics for each new database partition.

Step 5. Perform a final validation

To validate your environment:

1. Navigate to the */db2home/bcuaix/IBMredist* directory:
2. Run the postcheck script:

   ```
   ./redist_postcheck.ksh database partition-group
   ```

   Each execution of the postcheck script generates a unique output file such as redist_postcheck_ *database*_20120306045612.out in the directory in which you ran the script.

   The purpose of the postcheck script is to:

   - Compare output from the precheck and postcheck scripts to ensure that tables, table spaces, and other objects and configuration settings are the same.
   - Compare the number of pages that are used by table spaces across all database partitions to verify an even distribution of data
   - Confirm that you rebuilt indexes and re-created replicated MQTs

Compare output from the precheck and postcheck scripts to that tables, table spaces, and other objects and configuration settings are the same.

Monitoring and troubleshooting data redistribution

You can monitor the progress of the data redistribution utility in two ways: through the generated event logs and through the `LIST UTILITIES` command. You might also require the log files that are referenced here if you must contact IBM Support.

Redistribution event log files

Information about each processed table is logged in a single redistribution event log file in the *bcuaix*`/sqllib/redist` directory, where *bcuaix* is the instance owner's home directory. The event log file name is *dbname*`.`*dbpartgroupname*`.`*timestamp*`.log`.

The event logs provide general information about the redistribution operation, such as the old and new distribution maps. The logs also provide information about each table that you redistributed, including the following information:

- The indexing mode being used for the table
- An indication of whether the table was successfully redistributed
- The starting and ending times for the redistribution operation on the table

An example of an event log file follows:

```
                Data Redistribution Utility :
The following options have been specified :
Database partition group name : PDPG
Data Redistribution option : U
Redistribute database partition group : uniformly
No. of partitions to be added : 8
List of partitions to be added :
17
18
19
20
21
22
23
24
No. of partitions to be dropped: 0
List of partitions to be dropped:
Delete will be done in parallel with the insert.
The execution of the Data Redistribution operation on :

Begun at  Ended at  Table (poolID;objectID)

________  ________  ______________________
13.51.45            "PERFPOL1"."LINEITEM" (-6;-32764)
          14.26.51  "PERFPOL1"."LINEITEM" (-6;-32764)
14.26.51            "PERFPOL1"."LI_MQT" (10;17)
          14.34.09  "PERFPOL1"."LI_MQT" (10;17)
```

The `poolID` value of `-6` indicates that the table is range partitioned.

LIST UTILITIES command

Use the **LIST UTILITIES SHOW DETAIL** command to show the progress of the redistribute command. Sample output follows:

```
ID                                = 1
Type                              = REDISTRIBUTE
Database Name                     = BCUDB
Partition Number                  = 17
Description                       = PDPG UNIFORM ADD NODES
COMPACT ON DATA BUFFER SPACE REUSE RECORD LEVEL INDEXING
MODE DEFERRED
Start Time                        = 04/23/2012
08:34:36.851796
State                             = Executing
Invocation Type                   = User
Progress Monitoring:
   Estimated Percentage Complete = 100
   Summary:
      Total Work                  = 8618463082
      Completed Work              = 8609202192
      Total Number Of Tables      = 25
      Tables Completed            = 16
      Tables In Progress          = 1

   Current Table 1:
      Description                 = "BCUAIX
"."CATALOG_SALES"
      Total Work                  = 4455019151 bytes
      Completed Work              = 4347101184 bytes
      Start Time                  = 04/23/2012
08:39:45.899072
```

Conclusion

As the volume of data increases, the demands on data warehouses to accommodate that growth increases in lockstep. This paper describes not only how the IBM Smart Analytics System allows you to scale out your data warehouse, it also provides a clear set of instructions for redistributing the data to take full advantage of that expansion—and at the same time minimizing the effect of these activities on your business.

Best practices

- Download and use the scripts that are made available with this paper to help plan and prepare for database expansion and data redistribution in your environment.
- Redistribute data as a non-logged and non-recoverable operation with index rebuild deferred. This approach minimizes the disk space that is required and helps maximize the performance of the redistribution utility.
  - Expand the database separately from redistributing data. Separating the tasks allows time to address any expansion-related issues without affecting the timeline for data redistribution.
  - Complete a test run in a non-production environment to validate the steps, commands, scripts, and process that you intend to follow in production.
  - Increase the utility heap and sort heap memory allocation for the duration of the redistribution process.
  - To avoid locking, redistribute data during a maintenance window.
- Ensure that your backup infrastructure continues to support your recovery time objectives (RTOs) after expansion.
- Reduce the HWM to align the number of pages that are used by each table space across all database partitions and reduce the elapsed backup time on the original database partitions.
- Compare output from the precheck and postcheck scripts to ensure

that tables, table spaces, and other objects and configuration settings are the same.

Appendix A. Estimating the project timeline

You can gather much of the information that you need to estimate the time to complete the database expansion and redistribution of data by using existing metrics for the database and some basic guidelines. Use Table 1 as a guide for collecting the information.

Task	Description	Time (min.)
Add new database partition	Allow 10 minutes for each data node.	n
Alter TEMP16K table space	Allow 10 minutes to complete this across the entire database.	n
Back up database	Use precheck script output to help derive this value.	n
Redistribute data	Use the estimate in the precheck script output.	n
Rebuild indexes	Assume a processing speed of approximately 200 MB/sec. for systems based on Power 550 processor increasing to approximately 600 MB/sec for systems based on the Power 740 processor. Estimate the time required to rebuild all indexes based on the size of indexes as per the precheck output. Divide the size of indexes for the entire database partition group by the number of database partitions per data node, and then divide by the processing speed. For example: (500GB *1024) / 4 / 200 / 60	n
Re-create and refresh replicated MQTs	Reference the existing time frame that is allowed for an MQT refresh.	n
Reduce to HWM	Reducing each table space to the HWM is a relatively quick operation to complete before initiating the backup operation; typically minutes. This is an asynchronous task, and you should use the monitoring functions to verify that the task is complete.	n
Back up database post redistribute	Use precheck script output to help derive this value.	n
Collect statistics (issue the `RUNSTATS` command)	Reference the existing elapsed time for your `RUNSTATS` maintenance jobs.	n
Reclaim MDC extents	Perform this for MDC tables only. Elapsed time will be in seconds for each MDC table.	n
	Total elapsed time for expansion and redistribution project	**n**

Table 2. Elapsed time for a database expansion and data redistribution project

Appendix B. Restoring the database to a pre-redistribution state

Use the following example as a guideline in restoring the database to a pre-redistribution state. It is assumed that you followed the recommendations in this paper:

- You took a full database backup after adding all new database partitions to the database.
- No logged transactions were issued following the pre redistribute backup as they would not be included in the restore.
- You complete the redistribution as a non-logged operation.

To restore the database to a pre-redistribution state:

1. Restore the database from the backup image. An example follows:

```
db2 terminate
db2start
db2_all '<<+0< db2 RESTORE DATABASE BCUDB USE TSM TAKEN AT
20110715124435 into BCUDB'
db2_all ';<<-0< db2 RESTORE DATABASE BCUDB USE TSM TAKEN AT
20110715124435 into BCUDB replace existing'
```

2. Issue the `ROLLFORWARD` command. An example follows:

```
ROLLFORWARD DATABASE BCUDB TO END OF BACKUP AND STOP
DB2 ACTIVATE DB BCUDB
DB2 CONNECT TO BCUDB
```

3. Monitor the `RESTORE` command by issuing the following command:

```
DB2 LIST UTILITIES SHOW DETAIL
```

4. Check that the `/db2home/bcuaix/sqllib/db2nodes.cfg` file contains an entry for each database partition.

Appendix C. Scripts available with this paper

The `redist_precheck.ksh` and `redist_postcheck.ksh`.scripts that are referenced in this paper are available for download from the best practices page on the developerWorks site:

(http://public.dhe.ibm.com/software/dw/data/bestpractices/redist_scripts.zip).

Save the scripts on the administration (coordinator) node of your data warehouse. You must assign execute permission to each script. Each script generates an output file in the directory in which you place the script. For good housekeeping, create a separate directory to contain both of the scripts and their output files.

By default, the scripts query the catalog tables on the administration node only. This approach means that the scripts do not create any locks or affect any workload.

Appendix D. Building a list of commands for maintenance tasks

Following are some examples of how to generate a list of individual commands that you might need during the post-redistribution phase. You do not have to complete all the tasks in the following list, but they might be useful in your environment. These examples illustrate how you can use the system catalogs to obtain lists of tables for a database partition group.

- Generate a command to reclaim extents for each MDC table in the database partition group:

```
SELECT 'REORG TABLE ' || RTRIM(A.TABSCHEMA) || '.' ||
RTRIM(A.TABNAME) || ' RECLAIM EXTENTS ONLY;' FROM
SYSCAT.TABLES A, SYSCAT.TABLESPACES B,
SYSCAT.DATAPARTITIONS D WHERE A.CLUSTERED = 'Y' AND
A.TABSCHEMA = D.TABSCHEMA AND A.TABNAME = D.TABNAME AND
D.TBSPACEID = B.TBSPACEID AND DATAPARTITIONID = 0 AND
B.DBPGNAME = 'PDPG'
```

- Generate a command to rebuild all indexes for each table in the database partition group:

```
SELECT 'REORG INDEXES ALL FOR TABLE ' || RTRIM(A.TABSCHEMA)
|| '.' || RTRIM(A.TABNAME) || ' ON ALL DBPARTITIONNUMS'
FROM SYSCAT.TABLES A, SYSCAT.TABLESPACES B,
SYSCAT.DATAPARTITIONS D WHERE A.TABSCHEMA = D.TABSCHEMA AND
A.TABNAME = D.TABNAME AND D.TBSPACEID = B.TBSPACEID AND
DATAPARTITIONID = 0 AND B.DBPGNAME = 'PDPG' ORDER BY
B.TBSPACE
```

Generate a command to reduce the HWM for each table space in the database partition group:

```
SELECT 'ALTER TABLESPACE ' || TBSPACE || ' REDUCE MAX;'
FROM SYSCAT.TABLESPACES WHERE DBPGNAME = 'PDPG'
```

- Generate a command to issue the `RUNSTATS` command against each table in the database partition group:

```
SELECT 'RUNSTATS ON TABLE ' || RTRIM(A.TABSCHEMA) || '.' ||
RTRIM(A.TABNAME) || ' WITH DISTRIBUTION AND SAMPLED DE-
TAILED INDEXES ALL;' FROM SYSCAT.TABLES A,
SYSCAT.TABLESPACES B, SYSCAT.DATAPARTITIONS D WHERE
A.TABSCHEMA = D.TABSCHEMA AND A.TABNAME = D.TABNAME AND
D.TBSPACEID = B.TBSPACEID AND DATAPARTITIONID = 0 AND
B.DBPGNAME = 'PDPG'
```

- Generate a command to reset the compression dictionary for all tables in the database partition group:

```
SELECT 'REORG TABLE ' || RTRIM(A.TABSCHEMA) || '.' ||
RTRIM(A.TABNAME) || ' RESETDICTIONARY;' FROM SYSCAT.TABLES
A, SYSCAT.TABLESPACES B, SYSCAT.DATAPARTITIONS D WHERE
A.TABSCHEMA = D.TABSCHEMA AND A.TABNAME = D.TABNAME AND
D.TBSPACEID = B.TBSPACEID AND DATAPARTITIONID = 0 AND
B.DBPGNAME = 'PDPG'
```

This is not a mandatory step and is not needed in the vast majority of cases.

The compression dictionary is built on each new database partition, based on sampling. If the sampling does not produce compression statistics that are in line with those for the existing database partitions, you might want to use the `REORG TABLE RESETDICTIONARY` command on the new database partitions after completing the redistribution process. Compare the compression statistics in the output files for the precheck and postcheck scripts to determine whether there is a deviation.

Appendix E. Sample output for precheck script

Sample output from the precheck script follows. In this environment, 30 database partitions were added to an existing 266 database partition database.

```
#./redist_precheck.ksh BCUDB PDPG 30
================================================================
Starting precheck for database: BCUDB on Partition Group: PDPG
================================================================
Planning precheck enabled. For full precheck use FULL parameter
option. E.g. ./redist_precheck.ksh myDB PDPG FULL
Info: AIX Operating System
Info: DB2 Service Level for DB2
Info: v10.1.0.1 Service Level for DB2
Info: 1 Fixpack for DB2
Info: s120605 Build Level for DB2

# Database and Partition Group Statistics
Info: 52 Database Partition Groups exist
Info: 0 Database Partition Groups are in redistribute status
Info: 976215 of a total 1103849 GB is used for all table spaces
in the database BCUDB

# Space available on top 10 most full database container filesys-
tems.
/dev/lvfsp406   2529.25    110.79   96%    51586     1%
/db2fs/bcuaix/NODE0406
/dev/lvfsp404   2529.25    109.26   96%    51599     1%
/db2fs/bcuaix/NODE0404
/dev/lvfsp426   2529.25    109.04   96%    51597     1%
/db2fs/bcuaix/NODE0426
/dev/lvfsp438   2529.25    105.34   96%    51583     1%
/db2fs/bcuaix/NODE0438
/dev/lvfsp412   2529.25    102.10   96%    51596     1%
/db2fs/bcuaix/NODE0412
/dev/lvfsp410   2529.25    101.10   97%    51597     1%
/db2fs/bcuaix/NODE0410
/dev/lvfsp452   2529.25    100.67   97%    51596     1%
/db2fs/bcuaix/NODE0452
/dev/lvfsp420   2529.25     98.84   97%    51596     1%
/db2fs/bcuaix/NODE0420
/dev/lvfsp432   2529.25     95.90   97%    51583     1%
/db2fs/bcuaix/NODE0432
/dev/lvfsp442   2529.25     85.65   97%    51596     1%
/db2fs/bcuaix/NODE0442

Info: 167472 of a total 179983 GB is used for all table spaces in
the database partition group PDPG
Info: 266. Database Partitions exist for Partition Group PDPG
Info: 30 Database Partitions to be added to the database
Info: 296 database partitions in expanded database.

# Details of data to be moved
Info: 1534 GB of data exists in the database partition group -
the rest is indexes
```

```
Info: 1139 GB of Non MDC data exists in the database partition
group.
Info: 19918 GB equivalent of MDC data pages exists in the data-
base partition group (19155).

# Table space details.
Info: 205. DMS tablespaces and 0. SMS tablespaces exist in Data-
base Partition Group PDPG
Info: 160748MB is the largest tablespace size in Database Parti-
tion Group PDPG
Info: 11285MB is the average tablespace size in Database Parti-
tion Group PDPG

# Check table space utilization for skew across partitions.
DBPARTITIONNUM #TBSPACES   GB_TOTAL             GB_USED
-------------- ----------- -------------------- -----------------
2              205         231                  183
4              205         232                  184
6              205         233                  185
8              205         233                  184
10             205         232                  184
12             205         233                  184
14             205         232                  184
16             205         232                  184
18             205         232                  184
20             205         232                  184
22             205         231                  183
..(excerpt shown)

# Backup statistics for single database partition in current year
AVG_BACKUP_TIME_MINS MAX_BACKUP_TIME_MINS
-------------------- --------------------
                  29                 1177

# Compression Statistics
Info: 32. tables are compressed in Database Partition Group PDPG
Info: 6584675760 rows are compressed in Database Partition Group
PDPG
Info: 97. percent of rows are compressed in Database Partition
Group PDPG
Info: 16 percent of pages saved through compression in Database
Partition Group PDPG

# Runstats on tables
Info: 28. regular tables have Runstats statistics in Database
Partition Group PDPG

# Overview of tables in database partition group
TYPE CLUSTERED PARTITION_MODE TABLE_COUNT
---- --------- -------------- ---------------------------------
T    Y         H                                            8.
T    -         H                                           24.


# Table Statistics - Top 10 tables by fpages size
```

```
TABLE_NAME                                          CARD
TABLE_SIZE_GB        MDC MODE NPAGES                FPAGES
PAGE_RATIO           NUM_CELLS            NUM_CELLS_NOT_IN_DK
---------------------------------------------------- --------------
SAL.FACT_DAILY_SALES                                    3146227812
199 Y   H               13766688           293447232
21.31                19027                19027
SAL.FACT_DAILY_STORE_SALES
144904032                    5 Y   H                 2292498
146727360                64.00                     -
-
SAL.STORE_RETURNS                                      43321824
10 Y   H                 1656720             88481952
53.40                    -                    -
SAL.FACT_AUDIT
15084108                    0 Y   H                 2853162
45654336                16.00              7833599
7833599
SAL.FACT_DAILY_REVIEW                                 99240336
7 Y   H                 2292498             36683712
16.00                    -                    -
SAL.FACT_DAILY_STORE_REGION_REVIEW
101582208                    5 Y   H                 2292498
36683712                16.00                     -
-
SAL.ITEM_SALES                                   10764
0 -   H                   8424               9102834
1080.58                  -                     -
SAL.FACT_WEEKLY_STORE
1896983712                  81 -   H                 5452902
5456646                 1.00                      -
-
SAL.COST_CENTER                                                17082
0 -   H                   234                4043520
17280.00                 -                     -
SAL.COST_REF
10296                    0 -   H                   7722
2700360               349.69                     -
-

# Table Statistics - Top 10 tables by page ratio
TABLE_NAME                                          CARD
TABLE_SIZE_GB        NPAGES               FPAGES
PAGE_RATIO           NUM_CELLS            NUM_CELLS_NOT_IN_DK
---------------------------------------------------- --------------
SAL.FACT_DAILY_STORE_SALES
144904032                    5              2292498
146727360                64.00                     -
-
SAL.STORE_RETURNS                                      43321824
10              1656720             88481952               53.40
-                    -
SAL.FACT_DAILY_SALES                                    3146227812
199             13766688            293447232
21.31                19027                19027
SAL.FACT_AUDIT
15084108                    0              2853162
```

```
45654336                16.00              7833599
7833599
SAL.FACT_DAILY_STORE_REGION_REVIEW
101582208                    5              2292498
36683712                16.00                    -
-
SAL.FACT_DAILY_REVIEW                              99240336
7              2292498             36683712                16.00
-                    -
SAL.WEEKLY_POSITION                           81740880
10               638820              1898208                 2.97
-                    -
SAL.FACT_STORE_HISTORY                              151047936
5               401778               760032                 1.89
-

# Range Partitioned table statistics - Top 5 tables by number of
ranges in descending order
TABLE_NAME                              RANGES
----------------------------------- ------
SAL.COST_ACCOUNT_CENTER                 100.
SAL.COST_REF                            100.
SAL.ITEM_SALES                          100.
SAL.PRODUCT_SALES                       100.
SAL.FACT_WEB_SALES                      99.

# Index Statistics
Info: 58. indexes exist in Database Partition Group PDPG
Info: 114 GB of regular index space, and 0 GB of MDC index space
exists in Database Partition Group PDPG
Info: 49. partitioned indexes exist in Database Partition Group
PDPG

# The Top 5 tables with largest combined index count are:
TABLE_NAME                                             CARD
INDEX_ROWS           INDEX_LEAF_SIZE_GB
-------------------------------------------------- --------------
SAL.FACT_WEEKLY_STORE
1896983712           3793967424                   57
SAL.FACT_DAILY_SALES                                    3146227812
3153134790                   55
SAL.COST_CENTER                                                17082
34164                     0
SAL.STORE_GROUP
3744                 7488                     0
SAL.FACT_AUDIT
15084108             23643594                    0

# MDC index statistics - Top 10 tables by size of MDC indexes in
descending order
TABLE_NAME                                               SUM_NLEAF
-------------------------------------------------------- ----------
SAL.FACT_AUDIT                                                  106
SAL.FACT_DAILY_SALES                                 84
SAL.STORE_RETURNS                                     0
SAL.WEEKLY_POSITION                 0
SAL.FACT_STORE_HISTORY                      0
```

```
SAL.FACT_DAILY_STORE_REGION_REVIEW                  0
SAL.FACT_DAILY_REVIEW                        0
SAL.FACT_DAILY_STORE_SALES                          0

# Table counts
Info: 28 Non RP tables in Database Partition Group PDPG
Info: 101 ranges in Database Partition Group PDPG

# MDC and RP Statistics
Info: 8. MDC only tables exist in Database Partition Group PDPG
Info: 8. MDC only tables have Runstats statistics in Database
Partition Group PDPG
Info: 6. Range Partitioned tables with MDC exist in Database Par-
tition Group PDPG
Info: 6. Range Partitioned tables with MDC have Runstats statis-
tics in Database Partition Group PDPG

# Checking table STATUS.
Info: 0 tables are in an abnormal state PDPG
Info: 0. tables are in check pending state PDPG

# Replicated MQTs
Info: 0. Replicated MQTs exist

# Database configuration settings.
WARNING: Logarchmeth1 is configured. Please Set to OFF for Redis-
tribution
Info: 300000 UTIL_HEAP_SZ configuration setting is suitable for
redistribution
Info: 20000 SORTHEAP configuration setting is suitable for redis-
tribution
Info: 0. Event monitors running at this time
Info: 0. DMS table spaces do not have autoResize enabled.

# Estimate time for redistribute command to complete.
INFO: AIX processor determined.
INFO: 300 transfer rate determined.
INFO: 8 minutes is needed to process 101 additional ranges.
INFO: 120 minutes is total estimated time for redistribution of
data in this Database Partition Group.
================================================================
Database Redistribute Pre Check complete for BCUDB.
Check ./redist_precheck_BCUDB_PDPG_20120610042913.out for details
================================================================
```

Further reading

- IBM DB2 for Linux, UNIX, and Windows Version 10.1 Information Center: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1
- "REDISTRIBUTE DATABASE PARTITION GROUP command: topic: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/topic/com.ibm.db2.luw.admin.cmd.doc/doc/r0002069.html
- "Lowering the High-Water Mark of a Table Space" technote: http://www-01.ibm.com/support/docview.wss?uid=swg21006526
- "DB2 best practices: DB2 instance recovery for IBM Smart Analytics System" article: http://www.ibm.com/developerworks/data/library/techarticle/dm-1010db2instancerecovery/index.html
- DB2 Best Practices papers: http://www.ibm.com/developerworks/db2/bestpractices
- *Best Practices: Storage optimization with deep compression* paper: http://www.ibm.com/developerworks/data/bestpractices/deepcompression
- Download site for IBM Smart Analytics System and InfoSphere Balanced Warehouse documentation (requires registration and approval): https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=idwbcu
- IBM Smart Analytics System validated stack pages:

  http://www-304.ibm.com/support/docview.wss?uid=swg21429594
- "Choosing partitioning keys in DB2 Database Partitioning Feature environments" article and download site for data skew estimating script:

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1005partitioningkeys

Contributors

Aleks Santars
Data Movement Developer, IBM

Alvin Law
Senior Managing Consultant, IBM

Christopher Braudy
Data Migration Consultant, IBM

Enzo Cialini
STSM, Chief Architect, Data Warehousing and Next Generation Analytics, IBM

Kenton Delathouwer
IBM Smart Analytics System, Core Development, IBM

Larry Pay
IT Specialist, WW Lab Services, IBM

Leslie I McDonald
Technical Editor, DB2, IBM

Mike Bhola
Data Management Consultant, IBM

Sean McKeogh
Technical Manager, Data Movement Utilities, IBM

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

Without limiting the above disclaimers, IBM provides no representations or warranties regarding the accuracy, reliability or serviceability of any information or recommendations provided in this publication, or with respect to any results that may be obtained by the use of the information or observance of any recommendations provided herein. The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any recommendations or techniques herein is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Anyone attempting to adapt these techniques to their own environment do so at their own risk.

This document and the information contained herein may be used solely in connection with the IBM products discussed in this document.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.