IBMr

IBM® Smart Analytics System

Best Practices

Building a Recovery Strategy for an IBM Smart Analytics System Data Warehouse

Dale McInnis
IBM DB2 Availability Architect

Garrett Fitzsimons
IBM Smart Analytics System - Best Practices

Richard Lubell
IBM Smart Analytics System - Information Development

Issued June, 2010

Building a Recovery Strategy for an IBM Smart Analytics System Data Warehouse........ 1
Executive summary 3
Introduction 4
Planning a recovery strategy 5
Recovery point objectives and recovery time objectives........ 6
Physical data warehouse design........ 7
Data warehouse load method 8
Backup size and throughput capacity for recovery 9
Implementing a recovery strategy........ 12
System configuration........ 12
TSM 13
Backup guidelines........ 16
Recovery guidelines........ 18
Implementation scenario for a recovery strategy........ 21
Sample backup scenarios 22
Sample recovery scenarios........ 26
Administration tools and tasks 30
Conclusion 32
Appendix A. Configuration of test system 33
Appendix B. Configuring a DB2 instance for TSM........ 34
Appendix C. Overview of DB2 utilities........ 36
Further reading........ 38
Contributors........ 39
Notices 40
Trademarks 41

Executive summary

This paper is targeted at those involved in planning, configuring, designing, implementing, or administering a data warehouse based on DB2® Database for Linux®, UNIX, and Windows® software. In particular, this paper focuses on the IBM ®Smart Analytics System environment with configurations based on System x® and Power Systems™ servers.

The focus of a backup and recovery plan should be the recovery strategy. Identifying and planning for the recovery scenarios that are most likely to occur are the key factors in determining the speed with which you can recover from data loss or corruption. The goal of the recovery strategy should be to identify what has gone wrong and correct it by using the appropriate granular backup. A backup strategy should concentrate on online table space backups, which are more flexible than the traditionally larger full database backups. The backup strategy should also identify and back up active data, thereby reducing the need to continually back up inactive data. For maximum protection and efficient recovery of data, begin planning your database recovery strategy at project inception and carry it through each stage of the project: procurement, sizing, configuration, database design, data load design, and development.

Using the recommendations in this paper will help you develop a strategy that meets current needs but is also scalable and flexible so that it meets future needs. You can see the impact of a poorly designed backup and recovery solution not only when you require a recovery, but also in the performance of the system as a whole. When you use data warehouses which provide input to strategic decision-making and operational intelligence systems, you must be able to run backup processes concurrently with business workloads, without negatively affecting those workloads.

The IBM Smart Analytics System product incorporates best practices guidelines in building a stable and high-performance data warehouse environment. This paper builds on this approach by recommending that you balance backup processes across all database partitions in your system.

Introduction

This paper covers planning a recovery strategy for an IBM Smart Analytics System data warehouse.[1] A backup strategy will be a natural output of this planning. The focus is on recommendations. An IBM Smart Analytics System usage scenario is provided; you might have to modify commands that are used in this scenario for your environment. To use this paper, you should have a working knowledge of DB2 software.

A recovery strategy helps to ensure continuous availability of data. Defining your recovery objectives will dictate your backup processes. Planning backups and recovery should begin early and concentrate on recovery speed while maintaining an efficient, flexible, and balanced backup strategy.

This paper does not cover disaster recovery or operating system recovery; these topics are explored in other resources. Also, disk failures, which you can manage with RAID, and server failures, which you can manage with high availability architecture, are not covered here. This paper also does not cover offline backups, instance-level recoveries, and IBM FlashCopy® as components of an overall strategy.

The first section of the paper describes the concept of recovery objectives and considers the design of the data warehouse and load methods in planning a recovery strategy. This section highlights how decisions that you make early in the data warehouse project affect the recovery strategy and what the best practices guidelines are in this area.

The second section of the paper sets out recommendations on how to use configuration as well as backup and recovery techniques to implement a backup and recovery strategy that is balanced and focuses on the stated recovery objectives of your environment.

Following these sections is a sample implementation scenario for a backup and recovery strategy, an overview of monitoring, and appendixes on the configuration of the test system, configuring a DB2 instance for TSM and DB2 utilities.

[1] *The IBM Smart Analytics System is an evolution of InfoSphere Balanced Warehouse. They are based on the same storage and database design principles. All content in this paper, applies to IBM Smart Analytics System configurations. In most cases, the content also usually applies to custom data warehouses that are based on similar design principles, although some changes might be needed depending on the specific environment.*

Planning a recovery strategy

Many environments have a backup strategy that highlights the backup schedule, tape location, archiving, and retention policy without addressing recovery scenarios. A scenario-driven recovery strategy should define your backup planning. This section examines how to plan a recovery strategy and how decisions that you make during the design and development of a data warehouse affect how you back up data and how long a recovery process takes to run. You should review the effect of decisions on the recovery strategy at all stages of the implementation of a data warehouse.

These key recommendations are used as input to the recovery strategies that are described throughout the paper:

- Using **full table space backups** rather than full database backups. (Recovering a database from full table space backups became possible with the introduction of the **RESTORE REBUILD** command in DB2 9.5 software for Linux, UNIX, and Windows.) You can choose the frequency of full table space backups for both active and inactive data. This approach compensates for the added disk space and processor usage of including recovery metadata in the table space backup image.

- Using **online table space backups** of active data rather than offline database backups. Online table space backups support 24-hour data availability needs.

- Using **frequent full table space backups** rather than less frequent full backups and incremental backups in conjunction. Frequent full table space backups prioritize recovery speed over backup speed because fewer and smaller backup images and fewer transaction logs are required to restore lost data.

- Using **IBM Tivoli® Storage Manager (TSM)** software to assist in managing the backup and recovery process. Using TSM is recommended over taking local disk backups because integration with DB2 software reduces the complexity of managing backup images and transaction logs.

Planning a backup and recovery strategy should involve the following considerations:

- Recovery point objectives and recovery time objectives

- Physical data warehouse design

- Data warehouse load method

- Backup size and throughput capacity for recovery

Recovery point objectives and recovery time objectives

Planning a recovery strategy should begin with a comprehensive set of requirements. These should include the maximum data loss that is acceptable, the point at which a recovery scenario is triggered, and the time that is required for each recovery scenario.

Document your recovery objectives and use them as reference points when designing your backup strategy.

Recovery point objective (RPO) refers to the maximum amount of data loss that is acceptable in a recovery scenario. This value affects how you configure and manage your transaction logs and how you approach your backup strategy. This value might be different for a disaster recovery scenario.

Do not plan on meeting your RPO using Point in Time (PIT) recovery. PIT refers to the earliest moment to which you can recover a database or table space to a consistent state. PIT recovery of a single table space in an individual database partition is not possible in a database partitioned environment. Recovering a single table space to the same point in time across the database would require rolling forward to the ends of logs for all database partitions. If you must perform a PIT recovery, you must do it as a full database recovery.

Recovery time objective (RTO) refers to the time required to complete the recovery process. An RTO should state how you calculated the objective. The calculation is based on the following factors:

- When the problem is diagnosed and logged as an issue
- When the logged issue has been researched and accepted as a problem
- When the recovery procedure begins

Determining the RPO and RTO depends on the demands of the data warehouse as decided in the planning stages. In some environments, maintaining uptime, as specified by the RTO, might be more critical than suffering a certain amount of data loss, as specified by the RPO. For example, in a single-line-of-business data warehouse where a table space outage might render the data warehouse unusable, RTO would take priority. However, RPO might be more important in an enterprise data warehouse supporting multiple line-of-business feeds, where a single table space outage might impact only a small number of business users.

Reconcile your recovery objectives with the design and configuration of your data warehouse to ensure that they are achievable.

The RTO and RPO represent the expected service-level objectives. The more aggressive the RPO and RTO values, the higher the costs in hardware, tapes, and labor. Plan your strategy so that you are comfortable with the stated objectives, and reconcile these objectives against your environment to ensure that you can achieve them. For example, ensure that the hardware that is available can support the recovery times that you expect. Also, the earlier in the project cycle that you know the objectives, the more likely it is that you can influence sizing and configuration of hardware to meet these objectives.

Physical data warehouse design

The physical architecture of a data warehouse database must directly support the backup and recovery strategy. The architecture must help ensure good performance for the data load and applications but still provide backup and recovery options.

For example, a data warehouse with all tables and indexes in just a few table spaces is not conducive to a best-practices backup and recovery strategy due to the following factors:

- Poor separation of active from inactive data
- Slower and larger table space backups
- Slower table space recovery speed

Database partitioning distributes rows in a table across many database partitions to form a single large database. This provides advantages in scalability, load balancing, I/O, and recovery. For added flexibility, you can also use range partitioning for tables, resulting in subdivided table spaces both within and across database partitions.

Note: In this paper, a *full* table space backup refers to a backup for a table space that stretches across several database partitions. However, a table space failure or a table space recovery might refer to failure or recovery of a table space that is on a single database partition. Consider the context of the instructions when determining the type of the table space in question.

Identify pressure points and weaknesses in your environment because these are the most likely areas to fail, prompting a recovery.

Other considerations for physical data warehouse design are as follows:

- **Active and inactive data.** The focus of the backup strategy should be active data: “hot” data that is frequently written and “warm” data that is updated less often. Separate active data from inactive “cold” data and place each into separate table spaces, then back up table spaces that contain active data regularly between full database backups.

  When cold data is excluded from regular backups, the time required to complete your backups decreases because fewer backup processes are required. Targeting active data therefore also reduces backup traffic, facilitating online backups and thereby increasing system availability.

- **Range partitioning.** Separate range-partitioned data into individual table spaces. This technique can further separate active from inactive data and decrease backup and recovery time.

- **Referential integrity.** If tables are referenced across table spaces, restoring one particular table space might also require you to restore a different referenced table space, increasing recovery time. To avoid this situation, consider grouping smaller tables that reference each other into the same table space.

- **Compression.** Compressed tables do not benefit significantly from backup compression. Also, backup compression increases processor utilization and backup times significantly. Therefore, you should not plan to use backup compression where table data is already compressed or if you have workload concerns.

- **Staging tables.** Placing staging tables into their own table spaces allows you to exclude the tables from backups, thereby reducing time for both backup and recovery. Staging tables are used for loads, so if a failure occurs, you can reload the data into the tables. Do not back up staging tables or the temporary table space.

- **Local and global indexes**. Local indexes, which were introduced in DB2 9.7 software, allow for faster data load and attach of data partitions.

  If you use a global index, back up the global index associated with the targeted data table spaces. You must restore or rebuild the global index when recovering a table space, even when recovering a table space on a single database partition. Recovering data includes recovering associated indexes; therefore, index table spaces should follow the same backup schedule as data table spaces.

Data warehouse load method

You should include recovery planning in the data load design phase. The design of the data load (ETL or ELT) processes and maintenance of metadata both have a significant effect on the design and execution of the backup and recovery strategy. Recommendations are as follows.

- **Avoid loads directly into the target table.** Avoid using the load utility (which you run by issuing the **LOAD** command) to load data directly into data warehouse tables, unless the volume of data is low. If you load data without logging, the table space is placed in a non-recoverable state. However, setting the **COPY** parameter of the **LOAD** command to YES to trigger a table space backup is not a preferred solution because concurrently running a load and a backup can cause processor load issues. Instead, load data into a staging table before attaching the data partition to or inserting the data into the data warehouse. In addition, take advantage of any features provided by the ETL tools, such as the ability to perform parallel inserts.

- **Coordinate backups with the data load.** Coordinating data load processes and online backups avoids exclusivity and locking issues, specifically where non-logged data loads might fail. Additionally, your backup scripts should check whether backups are completed successfully and retry the backups if they fail.

- **Do not use recovery to correct load errors.** Using table space recovery to correct errors in the data load process will incur downtime. This is a slow and counterproductive way of recovering from application errors, which should be corrected at source.

Avoid using PIT recovery for ETL applications because the required offline backups are inefficient.

- **Maintain ETL metadata.** Most ETL tools and applications maintain metadata that contains details on the data load. Use this information to determine what tables and table spaces are hot, warm, or cold with regard to data changes and dynamically focus the backup schedule on those table spaces that are most active. Work with your ETL team to understand and influence what metadata is maintained.

A simple example of an ETL metadata table that shows when a table was last written to follows:

ETL run number	Execution start time	Execution stop time	Table	Rows inserted	Rows updated	Rows deleted
1001	20100402191702	20100402200002	Inventory	230012	0	0
1002	20100402194547	20100402195847	Sales	1900221	0	0
1003	20100402195632	20100402195711	Customer	122	12	0

Figure 1. Example of an ETL metadata table

Rolling out data from a data warehouse

Assume that data will at some point be loaded in error. The simplest mechanism for unloading a wrong, incomplete, or corrupt file is to delete the data that was loaded. However, deleting millions of rows can cause problems such as full transaction logs and escalated locks which might compromise the business workload and the backup and recovery strategy.

If you implemented range partitioning or multidimensional clustering, use the DELETE FROM … WHERE statement to target reasonably-sized blocks of data at an appropriate time at the data partition level, ensuring that the number of transactions being logged is minimized. Avoid using this statement in other situations because it might cause row-level locking and increase the volume of your log files greatly, thus unacceptably increasing your recovery time. Instead, use one of the following methods:

- Use the **LOAD** command with the **REPLACE** parameter to load from an empty file into a detached data partition.

- Logically mark the rows as deleted. You can then delete them in manageable chunks over time.

- Move valid data to a new table, create indexes, drop the old table, and rename the new table.

- If you are using DB2 9.5 software, use the ALTER TABLE statement, as shown in the following example, to delete the rows in the table and reload the data:

  ```
  alter table test.mytable activate not logged initially with empty table
  ```

- If you are using DB2 9.7 software, use the TRUNCATE TABLE statement to delete all rows from the table. The following example empties an unused inventory table regardless of any existing delete triggers but preserves its allocated space for later reuse.

  ```
  truncate table inventory reuse storage ignore delete triggers immediate
  ```

Backup size and throughput capacity for recovery

A recovery scenario is likely to involve a table space recovery from tape and archive log recovery from tape or disk. The recovery process can be broken down into the steps of reading the data from tape, transferring the data through a LAN or network attached storage to the target system, and writing the data to disk. Understanding these constituent parts assists in your calculations of size and throughput.

Backup image size. To understand the capacity of your system to back up and transfer data, you must determine the size of your backup images. Estimate the size of a table space backup image by multiplying the number of pages that are used in the table space by the size of each data page. This estimate will be in the upper range of the actual backup image size.

Use the following statement to estimate the size of a backup image:

```
select tablespace_name, snapshot_timestamp, min_recovery_time,
(sum(used_pages+pending_free_pages)* page_size)/(1024*1024) space_used_mb
from table(snapshot_tbs_cfg(' ',-1))
where tablespace _type = 0
group by tablespace_name, snapshot_timestamp, min_recovery_time, page_size
```

When estimating the size of an offline backup image, account for the PENDING_FREE_PAGES value, which is the number of pages that become free on a commit or rollback of a transaction. In the previous example statement, the number of pending-free pages is added to the number of used pages to estimate the size of an offline backup image. Completion of an offline backup does not automatically refresh the PENDING_FREE_PAGES value. Force the release of the pending-free pages column by issuing a **LIST TABLESPACES SHOW DETAIL** command before you take the database offline to back it up. An online backup attempts to free pages that are in a pending state, so remove the PENDING_FREE_PAGES value if you are calculating the size of an online backup image.

If you specify the **INCLUDE_LOGS** parameter for the **BACKUP DATABASE** command for an online backup, the transactions that occur while the backup is running increase the size of the backup image. TSM uses a similar method for estimating the size of the backup image in order to estimate storage pool usage.

Throughput capacity. To calculate the throughput capacity, you must know the configuration of the storage manager that you are using, how much disk space is available, and how many tape drives are available, as explained in the appendixes. For example, in an IBM Smart Analytics System, IBM LTO4 tape drives have a throughput of 120 MB per second for compressed data. This equates to 7.2 GB per minute and 432 GB per hour. These are only specifications and represent maximum throughput capacity rather than actual throughput. When these tape drives are in operation, fibre cable connectivity and software processes affect actual throughput.

Standard tape cartridge capacity for the LTO 4 is 800 GB. The minimum storage requirement is one tape drive per data module, but one tape drive per database partition is preferred to increase recovery speed.

Deploy two tape drives per data module to increase recovery speed.

Estimate the recovery time per scenario based on the size of the table space or database to be recovered, and compare this time against the RTO. Take the following items into consideration:

- **Number of tapes.** The number of tapes depends on how you configure your storage manager and the number and type of backup images that the restore utility must access to complete the recovery.

- **Quantity of transaction logs**. Based on elapsed time between backups, logs are a rollup of transactions and their number typically reflects the level of load activity.
- **Type of backup used for recovery.** If you use a full database backup for a table space recovery, your recovery is slower because the restore utility must traverse the database backup image to find data for the table space to be recovered.

Implementing a recovery strategy

Balanced performance across all the database partitions in a data warehouse is a key design goal for all workloads, including user queries, data loads, backups, data extracts, and logging. Executing backups in parallel across modules helps ensure this balance.

Best practices recommendations for implementing a recovery strategy cover the following areas:

- System configuration
- TSM
- Backup guidelines
- Recovery guidelines

System configuration

Configure the system to help ensure balance, stability, and performance. Recommendations focus on the following areas:

- Backup throttling
- Transaction logging

Backup throttling

You can use throttling to help manage processor usage by utilities and by the backup processes Throttling does not limit I/O resources; consider their availability, because a backup is an I/O-intensive operation.

You can use the **util_impact_lim** configuration parameter to set the instance-wide limit on the effect that all throttled utilities can cumulatively have on the system workload. Allowed values are 1 – 100, where the value corresponds to a percentage of workload. For example, a value of 20 limits utilities to affecting the system workload by no more than 20% in total, so if two backups are running with the same priority, they can affect the system workload by 10% each. The DB2 database manager continually monitors and adjusts the backup and other utility processes such that the effect on the system workload does not exceed the configured percentage.

The following statement demonstrates how to set the **util_impact_lim** parameter so that all utilities running concurrently cannot consume more than 33% of resources:

```
update database manager using util_impact_lim 33
```

You can also use the **util_impact_priority** configuration parameter to change the throttling priority of a utility while it is running, as follows:

- You can allow an individual utility to consume a greater percentage of the value assigned to the **util_impact_lim** parameter by giving the utility a higher priority.

- You can lessen the effect of a utility on resources by reducing its priority.

- You can remove throttling for an individual utility by setting the value of the parameter to `0`. The following example shows how to remove throttling for a utility with an ID of 92:

  ```
  set util_impact_priority for 92 to 0

  DB20000I The SET UTIL_IMPACT_PRIORITY command completed successfully.
  ```

You cannot apply utility throttling to the **RESTORE DATABASE** command.

Transaction logging

Transaction logs record all changes made to a database. You can use these logs to roll back or replay a transaction. In a data warehouse environment, it is recommended that you implement archive logging, which supports online backups, instead of circular logging, which does not support online backups.

How you configure the database parameters that determine the size and number of transaction logs has the following implications:

- Small transaction logs can reduce the risk of data loss in the event of failure because the logs contain fewer transactions. Large transaction logs increase the number of records that might be lost in the event of failure.
- Small logs can increase the number of active logs because large transactions might span multiple logs.
- Recovery speed is directly affected by the number of archived logs that must be retrieved.
- Large transaction logs use fewer system resources because less log switching takes place.

It is recommended that each database partition have 50 primary logs of 50 MB each (12,800 4 KB pages), for a total of approximately 2.5 GB of logs allocated per database partition. Modify the number and sizes of logs depending on your environment, but consider your data volumes and objectives.

Run frequent table space backups to minimize the number of transaction logs that are required in a recovery scenario.

Archiving transaction logs on a regular schedule can limit data loss. For example, if the recovery point objective is 60 minutes, manually archive the logs every 30 minutes so as to minimize the risk of losing a log file in the event of failure.

TSM

TSM, or Tivoli Storage Manager, is an IBM software tool that protects data from hardware failures and errors by storing data on offline and offsite storage. You should use TSM to manage the storage backup images, log files, operating system files, source data, scripts, and other important files associated with the data warehouse. You can also use TSM to schedule the execution of backup scripts.

TSM and DB2

By configuring the DB2 software **LOGARCHMETH1** configuration parameter to “TSM”, you can automatically direct archived log files to TSM rather than to a local disk. You can also direct backups to TSM by specifying the **USE TSM** parameter for the **BACKUP** command. When a backup takes place, data streams directly to TSM using a TSM connection, and data is saved to a storage pool that can be on disk or tape, depending on the configuration and policies that you implement. Opening multiple I/O streams creates parallelism.

In a recovery scenario, be aware of other workloads running and the time required for tape cartridges to be mounted. If parts of a recovery are running in parallel, queuing delays might occur when two streams are seeking data that is on the same tape.

TSM and storage

Install TSM client software on each administration and data module and then configure the DB2 software to archive files to and retrieve them from TSM. The external TSM server must be configured with both disk and tape storage media. RAID5 is recommended for TSM disk storage.

TSM reclaims disk space when data has been written to tape or has become obsolete as per the configuration. Because restoring from tape is more time-consuming than restoring from disk, configure TSM to retain on disk only the data that is most likely to be used in a recovery scenario. You should send full online database backups directly to tape, with archiving offsite for disaster recovery (DR) purposes.

Specifying the **INCLUDE LOGS** parameter with the **BACKUP DATABASE** command to include the logs with the database backup is recommended because it helps ensure that you can restore to a consistent state from a single image. When using this tactic, both the database backup and include log backup processes might attempt to access the same tape drive at the same time. To avoid a deadlock occurring during a full online backup, create two TSM tape pools: one for backups and one for logs.

TSM will experience problems servicing DB2 requests if disk and tape capacity on the TSM server is reached. Configure the **failarchpath** parameter so that logs are archived to local disk space on the data module in the event of TSM failure due to disk, tape, or network unavailability.

When rolling forward through the logs, the restore process first looks in the active log directory. If the log file is not found there, the path that the **overflowlogpath** configuration parameter points to is checked. If the log file is still not found, the process refers to the log archive path, which in this case is TSM. To speed up this process, pre-populate the **overflowlogpath** location with the log files needed by the recovery process. Use the **db2adutl** command to determine which logs are required on each database partition, as detailed in the recovery scenario.

TSM and LAN-free technology

Implementing LAN-free technology is recommended because it removes backup and recovery traffic from the corporate network by delivering the data directly between the storage area network device and the TSM tape device in both directions, increasing recovery speed.

LAN-free is suited to transferring large amounts of data such as backups. Short transfers or small files use LAN-free less efficiently and for this reason it is not recommended for use in archiving transaction logs. However, you can configure TSM to archive logs over the corporate network and send backup images directly to tape via LAN-free communications. This process is detailed in Appendix B.

You should implement LAN-free technology when network throughput becomes an issue, such as when either the backup or recovery throughput is insufficient to meet service-level objectives. This often occurs when backup traffic per database partition reaches 100 GB per hour.

Configuring TSM

A TSM server can scale to support hundreds of databases. In the case of a backup strategy that covers multiple data modules, each with four or eight database partitions, the key considerations for TSM configuration are as follows:

- Number of connections to TSM
- Number of parallel backups
- Number of I/O sessions per backup
- Number of tape drives per data module

Calculate the number of connections you will need given your environment and backup strategy. For example, assume that an online backup of a table space across eight database partitions requires eight connections. If you include logs in the backup image, a further eight connections are required, giving a total of 16. If your TSM server has the capacity, back up using multiple parallel sessions. If each table space backup uses two sessions, the total number of connections to TSM doubles from 16 to 32. Ensure that the number of connections can be supported by your particular implementation of TSM. Similarly, when backing up to tape, avoid starting more sessions than there are tape disk drives.

In most recovery scenarios, when you perform a restore, the most recent backups are used. Where applicable, retain the most recent backup images on the TSM disk storage pool to speed up any potential recovery process. Implement a policy in TSM for archived transaction logs where transaction logs for the last two backup cycles are retained on disk because this will help to speed up any recovery process. For operating system files, keep the last three to five versions and integrate this strategy with your operating system backup and recovery strategy.

For each data module, calculate the peak sizes of table space backups, transaction logs, and operating system files to back up per database partition, and then multiply by the number of database partitions for that data module. This will give you the details required for sizing your TSM storage requirements.

TSM and local disk backups

Archiving transaction logs and streaming backup images directly to TSM is recommended over using a local disk solution where data is backed up to local disk before being archived to TSM. Although a local disk solution increases backup speed, it also adds complexity at recovery time and can reduce recovery speed. For example, the recovery history file will record only the local disk location where you store backup images and transaction logs, without reference to TSM. Therefore, before you use the **RECOVER DATABASE** command, you must identify the files that you require and manually extract them from TSM and place them back on the local disk, increasing risk and reducing recovery speed

Similarly, using the **AUTO_DEL_REC_OBJ** parameter to specify that old images and log files are to be automatically deleted does not affect TSM because the DB2 software is not be configured for TSM. Using local disk backups necessitates manual deletion of old images and log files on both the local disk and in TSM, adding further risk.

Backup guidelines

Best practices considerations apply to the following areas:

- Table space backups
- Inactive data
- DB2 files in the operating system
- Full database backups

Table space backups

Table space backups are recommended, executed as follows:

- Perform a full online table space backup of hot and warm (active) table spaces twice a week, with relevant logs. You can also implement incremental table space backups for hot table spaces which receive a large volume of data daily. To enable incremental backups, set the **trackmod** configuration parameter to `Yes`. A full offline database backup is required to activate this parameter. Backups of warm data can be less frequent than backups of hot data as per your discretion.

- Perform a full online database backup of the catalog partition daily, logs included, unless the catalog partitions hold a large amount of business data. In that case, full table space backups of the catalog table space might be more appropriate.

Use online backups to minimize downtime.

- Back up operating system files as indicated in the section that follows twice a week in tandem with table space backups.
- Perform a full online database backup of database partitions quarterly.
- Perform a full table space backup after adding a table space.
- Schedule reruns of failed backups during off-peak periods to avoid imbalance.
- Include relevant global indexes in each table space backup. [2]

Perform the full backups of each table space in parallel across nodes, one database partition per node at a time. When you have completed the backup of the first table space, back up the second table space then the third until you have backed up all table spaces.

Measure the success of your backup strategy by whether the backups are completed with minimal effect on resources, rather than by elapsed time.

[2] *In DB2 9.5, global indexes are restored at the overall table level, not at the database partition level. Local indexes, which were introduced in DB2 9.7, allow for both the data and index table spaces to be restored at the database partition level.*

Minimize the effect of backups on the business workload by backing up one table space per database partition at a time. Also, run the backups in off-peak periods and when data loads and data extracts are not running.

Inactive data

Data in a data warehouse typically becomes inactive or "cold" when no longer subject to updates, inserts, or deletes after a period of time. Review the requirement to continue to include this data in a backup. Archive cold data backups.

Focus both the backup and recovery strategies on active data. This data might account for just 20 - 25% of the overall data warehouse.

DB2 files in the operating system

There are DB2 files in the operating system that you should include in any backup strategy, some of which are not included in the backup image. These files exist on each database partition. The following table shows a list of files that you should back up. All of these files are useful; essential files are indicated with a "Y" in the third column.

File	Description	Essential	Comment
`db2diag.log`	Diagnostic log, one per database partition	N	Useful in problem determination
`db2nodes.cfg`	Node configuration file	Y	Speeds recovery of setup
`db2rhist.asc`	Database recovery history file	N	Allows a specific history file to be referenced
`db2tschg.his`	Table space recovery history file	N	Allows a specific history file to be referenced

`dsm.opt`
`dsm.sys`
`.kshrc`
`userprofile`
	TSM configuration files. as outlined in Appendix A	Y	Speeds recovery of setup

Figure 2. DB2 files that reside on the operating system to back up

You should also backup the following non-essential items to speed recovery of setup:

- Backup, restore, and maintenance scripts
- The output of the following database manager settings, database settings and registry variables:
    - **get database configuration**
    - **get database manager configuration**
    - **db2set -a**
    - **db2look -d bcudb1 -f -o configuration.sql**

Full database backups

You should take a full online database backup quarterly, as stated in the preceding, and also where recommended during DB2 software upgrades or the addition of data modules or database partitions. Where an upgrade requires it, an offline backup may be performed.

When performing a full online database backup, continue the practice of maintaining balance. Back up the first database partition on each data module in parallel followed by the second, and so on.

Recovery guidelines

When a failure occurs and a recovery scenario is identified, you should follow the relevant documented and tested recovery scenario process. Regardless of the scenario involved, prioritize the restoration of active data. The best practices for recovery are as follows:

- Use table space recovery.
- Identify corrupt data.
- Classify the recovery scenario.

Use table space recovery

DB2 9.5 introduced the ability to rebuild all or part of a database by using table space backups, with no full database backup required. Each table space backup image now contains the structure of the entire database. After you rebuild the structure by using a selected table space backup, restore each table space in turn (or in parallel as preferred) on each database partition and roll forward to the end of each of the logs.

The advantages of a recovery strategy that is based on table space backups are as follows:

- Table space recovery targets only lost data, reducing I/O.
- Targeted table space recovery is faster than full database recovery.
- Table space recovery is more granular, which increases flexibility.

In a partitioned database environment, if a table space becomes logically or physically corrupt, it does so only within the data module on which the table space is located. The time that it takes to recover and roll forward a single table space on a single database partition from a table space backup is significantly less than the time that it takes to search through a full, incremental, or delta database backup (or a combination of those types of backups) and a large volume of transaction logs.

Use tools to speed up the recovery process. You can use the db2adutl command to retrieve logs for rolling forward while the restore is being processed.

In the case of a full database failure in a partitioned database environment, recover the catalog database partition first from the last full catalog database partition backup and then recover the remaining database partitions by using the table space backups. This process is shown in more detail in the sample scenario in the following chapter.

Identify corrupt data

Data corruption can occur due to failure within the hardware, operating system, or DB2 software layer. Use the DB2 **INSPECT** command and **db2dart** command to analyze the problematic database, table space, or table.

The following sample command reports on the integrity of the database architecture, including errors:

```
inspect check database results keep inspect.output
```

In this case, the output of the command is written to the `inspect.output` file. To format the contents of the file, use the **db2inspf** command.

First determine whether the data corruption exists on disk. If the corruption is in temporary objects or in-memory only, no recovery is required. If the corruption is local to a table, use the dropped table recovery process as described in DB2 documentation. If the corruption is confined to a single table space, a table space recovery will suffice. If the corruption is widespread, a full database restore might be required. Focus recovery on lost data only.

Classify the recovery scenario

Determine the sequence of table spaces to recover as defined by the recovery scenario. These are the recovery scenarios that you are most likely to encounter:

- **Table space loss.** Usually, table space losses are due to data corruption or storage failure. The quickest recovery method is to recover the affected table spaces rather than the database. Following best practices database design principles should minimize the number of table spaces to recover, with other table spaces remaining online and available to the user while the restore is in progress.
- **Table dropped in error.** To recover a dropped table, use the dropped table recovery feature. This feature is available for a table space if the DROP_RECOVERY column in the SYSCAT.TABLESPACES catalog view is set to Y for that table space. This feature is activated by default in a partitioned database environment. The process involves a restore and rollforward operation and an additional step in which the contents of the dropped table are output to a file that is compatible with the requirements of the **IMPORT** command. The file is used for later recovery of the table. In a partitioned database, the dropped table recovery feature is always on even if dropped table recovery is turned off for non-partitioned tables in one or more table spaces.
- **Database partition loss.** Database partition recovery might be required due to a storage failure, for example, a multiple-disk failure in an environment where RAID is not used. Database partition recovery involves restoring an individual database partition.
- **Catalog partition loss.** Catalog partition recovery might be required due to a storage failure; for example, a multiple-disk failure that exceeds the capacity of the RAID type. Catalog partition recovery involves restoring the catalog database partition, which you should back up more frequently than the other database partitions.
- **Application errors.** The preferred method of recovery from an application error is for the application to address the issue, for example, by using data unload.

- **Full database loss or alternative location restore.** The need for a full database or alternative location restore might arise from a disaster recovery scenario or a requirement to create a copy of the data warehouse on different hardware. You must restore the catalog database partition first, followed by all database partitions in parallel.

.

Implementation scenario for a recovery strategy

This section details a backup and recovery strategy for a sample data warehouse that was used for testing. The sample data warehouse consists of one administration module and two data modules, each with four database partitions, for a total of eight database partitions. The partitioned database was installed and configured in line with IBM Smart Analytics System best practices.

The sample data warehouse contains two database partition groups:

- A single-partition database partition group containing the catalog partition. This partition group contains a single table space that contains relatively small non-partitioned tables, primarily used for lookup purposes.

- A multi-partition database partition group containing all other database partitions. This database partition group was created across the eight database partitions and contains 17 table spaces. This database partition group contains the following tables:

    - Large dimension tables, such as customer and inventory tables, were created as hash-partitioned tables across all database partitions in the group.

    - Seven fact tables, which are both range partitioned using a date key and hash partitioned and are in individual table spaces. Separate table spaces are used for each index.

    - A number of materialized query tables (MQTs) in an MQT table space.

The data-load application used a stage-insert approach for loading data into the data warehouse, ensuring that all writes were logged.

Sample recovery scenarios

The following table summarizes the data warehouse recovery scenarios, the recovery methods and their effects, and the associated RPO and RTOs. These scenarios are based on the physical design, the data load method and the tested speed of data transfer for backup and recovery. For details of the commands and processes, see the appendixes.

Recovery Scenario	**Recovery Method**	**Impact**	**RTO**	**RPO**
Data loss in partitioned table	Restore the table space in the database partition, and roll forward to end of logs	Table space is offline	*n* minutes	Last transaction
Partitioned table space loss	Restore the table space in all database partitions, and roll forward to end of logs	Table space is offline	*n* minutes	Last transaction
Partitioned table loss	Restore the table space in all database partitions, and roll forward to end of logs	Table space is offline	*n* minutes	Last transaction
Database partition loss	Restore the database partition, and roll forward to end of logs	Data module is offline	*n* minutes	Last transaction

Catalog user table data loss	Restore the user table space, and roll forward to end of logs	Table space is offline	*n* minutes	Last transaction
Catalog user table space loss	Restore the user table space, and roll forward to end of logs	Table space is offline	*n* minutes	Last transaction
Catalog user table loss	Restore the table space, and roll forward to the PIT when the table was lost, perform the dropped table recovery process, and then rollforward to end of logs.	Table space is offline	*n* minutes	Last transaction
Catalog system table loss	Restore the database (catalog) partition, and roll forward to end of logs.	Database is offline	*n* minutes	Last transaction
Catalog partition loss	Restore the database (catalog) partition, and roll forward to end of logs.	Database is offline	*n* minutes	Last transaction
Partitioned database loss	Rebuild the entire database from the table space backups and roll forward to end of logs	Database is offline	*n* minutes	Last transaction

Figure 3. Recovery scenarios anticipated in test system

Sample backup scenarios

The backup strategy is based on the recommendations made in this paper. The strategy comprises a daily backup of the catalog partition, a full backup of all active table spaces on Sunday and Wednesday, and a full database backup every quarter. All backups are throttled at 33%. Examples using TSM are shown in this section.

The TSM client was installed on each data module as described in Appendix A. Archive logging was enabled and configured to use TSM by using the following command:

```
db2 update db cfg for bcudb1 using LOGARCHMETH1 TSM
```

where *bcudb1* is the name of the database to which the configuration change is being made.

Incremental backups were enabled by setting the **trackmod** configuration parameter to Yes as shown in the following example, and by performing a full database backup.

```
update database manager configuration using TRACKMOD YES
```

To verify that the transaction logs were being archived to TSM, the following command was issued:

```
db2pd -log -database bcudb1
```

Online daily database backup of the catalog partition

The administration module contains only one database partition. The command used to backup to Tivoli Storage Manager software (“TSM”) is as follows:

```
# Monday ~ Sunday
db2 -v "backup database bcudb1 on dbpartitionnums (0) online use tsm open 2
sessions util_impact_priority 33 include logs"
```

When the catalog partition is backed up, two I/O sessions are opened between the backup process and TSM. Because the **include logs** option is specified, three backup images are created: one per session and one for the transaction logs.

Querying TSM by using the db2adutl command

Use the **db2adutl** command to query, delete, extract and verify backup images and logs that you archive to TSM.

The following sample command verifies the integrity of the backup image of the catalog partition in the data warehouse.

```
db2adutl verify full database bcudb1 dbpartitionnum 0 taken at 20100420115405
```

Sample output is as follows. Note the timestamp because it is used in the delete command later in this section:

```
Query for database BCUDB1
Retrieving FULL DATABASE BACKUP information.  Please wait.

   FULL DATABASE BACKUP image:
  ./BCUDB1.0.bcuinst2.NODE0000.CATN0000.20100420115405.001, DB Partition Number: 0
  ./BCUDB1.0.bcuinst2.NODE0000.CATN0000.20100420115405.002, DB Partition Number: 0
  ./BCUDB1.0.bcuinst2.NODE0000.CATN0000.20100420115405.003, DB Partition Number: 0

Do you wish to verify this image (Y/N)? Y
Verifying file: ./BCUDB1.0.bcuinst2.NODE0000.CATN0000.20100420115405.001
####
Read 0 bytes, assuming we are at the end of the image
                     ./BCUDB1.0.bcuinst2.NODE0000.CATN0000.20100420115405.002
##
WARNING only partial image read, bytes read: 16384 of 16781312
Read 0 bytes, assuming we are at the end of the image
                     ./BCUDB1.0.bcuinst2.NODE0000.CATN0000.20100420115405.003
#####

Read 0 bytes, assuming we are at the end of the image
Image Verification Complete - successful.

Retrieving INCREMENTAL DATABASE BACKUP information.
  No INCREMENTAL DATABASE BACKUP images found for BCUDB1

Retrieving DELTA DATABASE BACKUP information.
  No DELTA DATABASE BACKUP images found for BCUDB1
bcuinst2@beluga002:~>
```

Important: The **db2adutl** command does not support single system view (SSV) backups at this point. You must therefore issue the command from the data module on which you took the backup.

Delete a backup image by using the **db2adutl** command, as shown in the following command:

```
db2adutl delete full database bcudb1 dbpartitionnum 0 taken at 20100420115405
```

Use the **db2adutl** command to delete old backup images according to a schedule, such as after the quarterly full database backups.

You can also use the command to query or extract transaction logs. The following command queries the logs associated with the catalog database backup:

```
   db2adutl query logs between S0000016.LOG and S0000019.LOG  database bcudb1
```

Sample output is as follows:

```
   Query for database BCUDB1
   Retrieving LOG ARCHIVE information.
Log file: S0000016.LOG, Chain Num: 1, DB Partition Number: 3, Taken at: 2010-04-02-11.10.16
Log file: S0000017.LOG, Chain Num: 1, DB Partition Number: 3, Taken at: 2010-04-02-11.10.19
Log file: S0000016.LOG, Chain Num: 1, DB Partition Number: 2, Taken at: 2010-04-02-11.10.20
Log file: S0000016.LOG, Chain Num: 1, DB Partition Number: 4, Taken at: 2010-04-02-11.10.21
Log file: S0000016.LOG, Chain Num: 1, DB Partition Number: 1, Taken at: 2010-04-02-11.10.22
Log file: S0000017.LOG, Chain Num: 1, DB Partition Number: 2, Taken at: 2010-04-02-11.10.22
Log file: S0000018.LOG, Chain Num: 1, DB Partition Number: 3, Taken at: 2010-04-02-11.10.22
Log file: S0000017.LOG, Chain Num: 1, DB Partition Number: 4, Taken at: 2010-04-02-11.10.23
Log file: S0000019.LOG, Chain Num: 1, DB Partition Number: 3, Taken at: 2010-04-02-11.10.24
Log file: S0000018.LOG, Chain Num: 1, DB Partition Number: 4, Taken at: 2010-04-02-11.10.25
Log file: S0000018.LOG, Chain Num: 1, DB Partition Number: 2, Taken at: 2010-04-02-11.10.26
Log file: S0000017.LOG, Chain Num: 1, DB Partition Number: 1, Taken at: 2010-04-02-11.10.27
Log file: S0000019.LOG, Chain Num: 1, DB Partition Number: 4, Taken at: 2010-04-02-11.10.27
Log file: S0000018.LOG, Chain Num: 1, DB Partition Number: 1, Taken at: 2010-04-02-11.10.28
Log file: S0000019.LOG, Chain Num: 1, DB Partition Number: 2, Taken at: 2010-04-02-11.10.28
Log file: S0000019.LOG, Chain Num: 1, DB Partition Number: 1, Taken at: 2010-04-02-11.10.29
```

The following command extracts the logs that are associated with the catalog database backup:

```
   db2adutl extract logs between S0000016.LOG and S0000019.LOG  database bcudb1
```

Twice-weekly online partitioned table space backups

Twice weekly, each table space is backed up in turn and in parallel across database partitions with only one backup process executed per data module at a time. The following commands, shown with sample output, back up the PDTS_IN table space, which is used for the fact tables and the associated PDTS_INI index table space on all database partitions. The following commands back up the first database partition on each data node in parallel:

```
   # Sunday (Full ~ Partitions 1 and 5)
   backup database bcudb1 on dbpartitionnums (1,5) tablespace(pdts_in, pdts_ini)
   online use tsm open 2 sessions util_impact_priority 33 include logs
   Part  Result
   ----  ------------------------------------------------------------------------
   0001  DB20000I  The BACKUP DATABASE command completed successfully.
   0005  DB20000I  The BACKUP DATABASE command completed successfully.
   Backup successful. The timestamp for this backup image is : 20100304122637

   # Sunday (Full ~ Partitions 2 and 6)
   backup database bcudb1 on dbpartitionnums (2,6) tablespace(pdts_in, pdts_ini)
   online  use tsm open 2 sessions util_impact_priority 33 include logs
   Part  Result
   ----  ------------------------------------------------------------------------
   0002  DB20000I  The BACKUP DATABASE command completed successfully.
   0006  DB20000I  The BACKUP DATABASE command completed successfully.
   Backup successful. The timestamp for this backup image is : 20100304123634

   # Sunday (Full ~ Partitions 3 and 7)
   backup database bcudb1 on dbpartitionnums (3,7) tablespace(pdts_in, pdts_ini)
   online  use tsm open 2 sessions util_impact_priority 33 include logs
   Part  Result
```

```
----  ------------------------------------------------------------------
0003  DB20000I  The BACKUP DATABASE command completed successfully.
0007  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful. The timestamp for this backup image is : 20100304124219

# Sunday (Full ~ Partitions 4 and 8)
backup database bcudb1 on dbpartitionnums (4,8) tablespace(pdts_in, pdts_ini)
online  use tsm open 2 sessions util_impact_priority 33 include logs
Part  Result
----  ------------------------------------------------------------------
0004  DB20000I  The BACKUP DATABASE command completed successfully.
0008  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful. The timestamp for this backup image is : 20100304125102
```

Quarterly and on demand full online database backups

Quarterly and on demand, a full online backup is achieved by first taking a backup of the catalog database partition. On completion, the first database partition on both modules is backed up in parallel, followed by the second, the third, and finally the fourth database partition. This helps maintain balance across all database partitions. The sequence of commands to back up the database and associated sample output are as follows:

```
db2 backup database bcudb1 on dbpartitionnum(0) online use tsm
Part  Result
0000  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful. The timestamp for this backup image is: 20100401114743

db2 backup database bcudb1 on dbpartitionnums(1,5) online use tsm
Part  Result
0001  DB20000I  The BACKUP DATABASE command completed successfully.
0005  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful. The timestamp for this backup image is: 20100401114912

db2 backup database bcudb1 on dbpartitionnums(2,6) online use tsm
Part  Result
0002  DB20000I  The BACKUP DATABASE command completed successfully.
0006  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful. The timestamp for this backup image is: 20100401115129

db2 backup database bcudb1 on dbpartitionnums(3,7) online use tsm
Part  Result
0003  DB20000I  The BACKUP DATABASE command completed successfully.
0007  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful. The timestamp for this backup image is: 20100401115603

db2 backup database bcudb1 on dbpartitionnums(4,8) online use tsm
Part  Result
0004  DB20000I  The BACKUP DATABASE command completed successfully.
0008  DB20000I  The BACKUP DATABASE command completed successfully.
Backup successful. The timestamp for this backup image is: 20100401115946
```

Sample recovery scenarios

The following sections illustrate the steps and related commands involved in applying various recovery scenarios to the sample data warehouse.

Recovering a table space on a single database partition

The first step is to issue the **RESTORE DATABASE** command, specifying the table space to restore and the database partition that is the target of the restore. The following sample command targets database partition 3, table space PDTS_IN and index table space PDTS_INI. The +3 in the db2 command specifies the third database partition.

```
db2_all "<<+3<db2 \"restore database bcudb1 tablespace (pdts_in, pdts_ini) online
use tsm taken at 20100401143616 replace existing \" "
```

When the restore has completed, issuing the following **ROLLFORWARD DATABASE** command with the **QUERY STATUS** parameter shows that database partition 3 is in rollforward pending state:

```
rollforward database bcudb1 query status
```

Sample output is as follows:

```
Rollforward Status
Input database alias                   = bcudb1
Number of nodes have returned status   = 9

Node  Rollforward Next log          Log files            Last committed transaction
      status      to be read        processed
----  ----------- -----------       ----------           --------------------------
0     not pending S0000000.LOG-S0000002.LOG  2010-04-01-09.35.31.000000 Local
1     not pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.37.000000 Local
2     not pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
3     TBS pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.44.000000 Local
4     not pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
5     not pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
6     not pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
7     not pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
8     not pending S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
```

The table spaces in this database partition must be consistent with the table spaces in other database partitions and, therefore, a rollforward to the ends of the logs is expected. On the database partition being recovered, the **ROLLFORWARD DATABASE** command is then issued for the table spaces:

```
rollforward db bcudb1 to end of logs on dbpartitionnum (3) tablespace (PDTS_IN,
PDTS_INI) online
```

Sample output from the command is as follows and shows that the rollforward status in no longer pending:

```
Rollforward Status
Input database alias                   = bcudb1
Number of nodes have returned status   = 1
Node number                            = 3
Rollforward status                     = not pending
Next log file to be read             =
Log files processed                   =  -
Last committed transaction     = 2010-03-04-13.53.03.000000 UTC
DB20000I  The ROLLFORWARD command completed successfully.
```

Recovering table spaces across all database partitions

In this example, the table space PDTS_IN and its associated index table space PDTS_INI are restored on all eight database partitions. To do that, a **RESTORE DATABASE** command is issued in a separate session for each database partition. The following example shows the command for the first database partition. The +1 in the **db2_all** command specifies the first database partition.

```
db2_all "<<+1<|| db2 \"restore database bcudb1 tablespace (PDTS_IN, PDTS_INI)
online use TSM taken at 20100304122637 replace existing\""
rah: omitting logical node 0

rah: primary monitoring process for db2 is 14932
beluga003: DB20000I  The RESTORE DATABASE command completed successfully.
beluga003: db2 "restore db bcudb1 ... completed ok
```

When you use a **RESTORE DATABASE** command to recover table spaces across all database partitions, the number of sessions that you can run in parallel depends on your environment and the number of database partitions and tape drives available.

After the table space is restored, a **ROLLFORWARD DATABASE** command is issued to roll forward to the ends of the logs. Based on twice weekly backup recommendations, no more than two and a half days of transaction logs should be required. The command is shown here:

```
rollforward database bcudb1 to end of logs on dbpartitionnum (1 to 8) tablespace
(PDTS_IN, PDTS_INI) online
```

Sample output from the command is as follows and shows that the rollforward status is no longer pending for all database partitions:

```
Rollforward Status
Input database alias                    = bcudb1
Number of nodes have returned status    = 8

Node number  Rollforward  Next log   Log files processed Last committed transaction
             status       to be read
-----------  -----------  ---------- ------------------  ------------------------
          1  not pending          -             2010-03-04-13.53.03.000000 UTC
          2  not pending          -             2010-03-04-13.53.06.000000 UTC
          3  not pending          -             2010-03-04-13.52.56.000000 UTC
          4  not pending          -             2010-03-04-13.53.04.000000 UTC
          5  not pending          -             2010-03-04-13.53.21.000000 UTC
          6  not pending          -             2010-03-04-13.52.59.000000 UTC
          7  not pending          -             2010-03-04-13.53.04.000000 UTC
          8  not pending          -             2010-03-04-13.53.03.000000 UTC
DB20000I  The ROLLFORWARD command completed successfully.
```

Recovering a catalog database partition

If the catalog partition is unavailable, the entire database is unavailable because the catalog and coordinator functions are offline. For this reason, it is recommended that a full online backup take place daily so that restoring the catalog partition can be as quick and simple as possible. The following command recovers the catalog database partition:

```
recover database bcudb1 to end of logs on dbpartitionnum (0)
```

The sample output from the command shows the log files used to perform the operation. Also note the status is not pending as the recover command performed both the restore and rollforward process:

```
Rollforward Status
 Input database alias                   = bcudb1
 Number of nodes have returned status   = 1
 Node number                             = 0
 Rollforward status                      = not pending
 Next log file to be read                =
 Log files processed                     = S0000000.LOG - S0000004.LOG
 Last committed transaction              = 2010-04-02-07.15.10.000000 Local
```

Executing the **LIST UTILITIES** command while the **RECOVER DATABASE** command is in progress shows the restore occurring:

```
list utilities show detail

ID                                 = 92
Type                               = RESTORE
Database Name                      = BCUDB1
Partition Number                   = 0
Description                        = db
Start Time                         = 04/02/2010 07:31:30.907354
State                              = Executing
Invocation Type                    = User
Progress Monitoring:
      Completed Work               = 134275072 bytes
      Start Time                   = 04/02/2010 07:31:30.907358
```

Executing the **LIST UTILITIES** command while the **ROLL FORWARD** command is in progress shows the roll forward occurring:

```
list utilities show detail

ID                                 = 93
Type                               = ROLLFORWARD RECOVERY
Database Name                      = BCUDB1
Partition Number                   = 0
Description                        = Database Rollforward Recovery
Start Time                         = 04/02/2010 07:31:58.121968
State                              = Executing
Invocation Type                    = User
Progress Monitoring:
   Phase Number [Current]          = 1
      Description                  = Forward
      Completed Work               = 13372055 bytes
      Start Time                   = 04/02/2010 07:31:58.121972
   Phase Number                    = 2
      Description                  = Backward
      Completed Work               = 0 bytes
      Start Time                   = Not Started
```

Recovering a user table space on the catalog database partition

If you must recover a user table space on the catalog database partition, it is recommended that you restore just the data that is lost instead of restoring the complete catalog database partition, which forces the database offline. To restore a user table space on the catalog database partition, first issue the RESTORE command, specifying the table space being restored. The following command restores the table space SDTS:

```
restore database bcudb1 tablespace (SDTS) online use TSM taken at 20100304041653
replace existing
```

When the RESTORE command completes, it will place the table space in a Roll forward pending state. The following command will roll forward to end of logs for the table space and returns it to a not pending state:

```
rollforward database bcudb1 to end of logs on dbpartitionnum (0) tablespace (SDTS)
online
```

Recovering an entire data module

To recover an entire data module from table space backups, the following process was required. In testing the first data module, which contains database partitions 1 through 4, was chosen for recovery. The following command was used to rebuild the data module by using a table space backup image:

```
recover database bcudb1 to end of logs on dbpartitionnum (1 to 4)
```

The command generated the following output:

```
Rollforward Status
Input database alias                    = bcudb1
Number of nodes have returned status    = 4

Node   Rollforward    Next log    Log files processed    Last committed transaction
Number status         to be read
------ ------------   ----------  --------------------  --------------------------
1   not pending         S0000000.LOG-S0000105.LOG  2010-04-02-07.15.09.000000 Local
2   not pending         S0000000.LOG-S0000106.LOG  2010-04-02-07.14.04.000000 Local
3   not pending         S0000000.LOG-S0000105.LOG  2010-04-02-07.14.04.000000 Local
4   not pending         S0000000.LOG-S0000106.LOG  2010-04-02-07.14.04.000000 Local
DB20000I  The RECOVER DATABASE command completed successfully.
```

Recovering an entire database

There are two methods available to recover an entire database. The first is from the quarterly or ad hoc full online database backups; the second is from the table space backups. The need to recover an entire database will most likely occur during a system upgrade. In such an event a full database backup should have been taken prior to the upgrade taking place and a restore from those backup images would take place. In the event of a recent full database backup not being available, table space backups would be used. This section will focus on a table space based recovery.

In the following example, the recovery is done by using table space backups. A single **RECOVER DATABASE** command is sufficient to search the recovery history file and restore to the ends of the logs. The following command recovers the entire database across partitions 0 through 8:

```
recover db bcudb1 to end of logs
```

Sample output is as follows:

```
Rollforward Status
Input database alias                    = bcudb1
Number of nodes have returned status    = 9

Node   Rollforward  Next log   Log files processed     Last committed transaction
number status       to be read
------  -----------  ---------  --------------------   --------------------------
0  not pending          S0000000.LOG-S0000002.LOG  2010-04-01-09.35.31.000000 Local
1  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.37.000000 Local
2  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
3  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.44.000000 Local
4  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
5  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
6  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
7  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local
8  not pending          S0000000.LOG-S0000003.LOG  2010-03-29-07.42.36.000000 Local

DB20000I  The RECOVER DATABASE command completed successfully.
```

Administration tools and tasks

There are several ways to view current and historical recovery events. The progress of a backup or recovery process can be monitored by using the **LIST UTILITIES** command. History files show what recovery events have taken place.

Monitoring utilities

Use the **LIST UTILITIES** command to identify what utilities are running, including the backup and restore utilities, and how they are progressing. The following sample output, which was produced by entering `list utilities,` shows that a backup is in progress on the test system that was used for the sample data warehouse:

```
ID                                 = 5
Type                               = BACKUP
Database Name                      = BCUDB1
Partition Number                   = 1
Description                        = online tablespace PDTS_IN...
Start Time                         = 03/29/2010 07:42:07.491717
State                              = Executing
Invocation Type                    = User
Throttling:
   Priority                        = 33
Progress Monitoring:
   Estimated Percentage Complete = 3

ID                                 = 5
Type                               = BACKUP
Database Name                      = BCUDB1
Partition Number                   = 5
Description                        = online tablespace PDTS_IN...
Start Time                         = 03/29/2010 07:42:07.638023
State                              = Executing
Invocation Type                    = User
Throttling:
   Priority                        = 33
Progress Monitoring:
   Estimated Percentage Complete = 4
```

Viewing history by using the SYSIBMADM.DB_HISTORY table

Query the SYSIBMADM.DB_HISTORY table to view history information about recoveries and backups, including when and how long they ran. You can also query the table to view information about logs.

The following sample statement queries recovery history:

```
selelct   dbpartitionnum, start time, end time,
       substr(COMMENT,1,20) comment, substr(LOCATION,1,20) location
from      sysibmadm.db_history
where     operation in ('R') and seqnum = '1' and dbpartitionnum <= 4
          order by start time, dbpartitionnum
```

Sample output is as follows:

```
DBPARTITIONNUM START_TIME     END_TIME       COMMENT              LOCATION
             0 20100401093833 20100401093843 RESTORE BCUDB1 WITH RF    -
             0 20100401102056 20100401102105 RESTORE BCUDB1 WITH RF    -
             2 20100401102056 20100401102654 RESTORE BCUDB1 WITH RF    -
             3 20100401102056 20100401102657 RESTORE BCUDB1 WITH RF    -
             1 20100401102057 20100401102655 RESTORE BCUDB1 WITH RF    -
             4 20100401102057 20100401102649 RESTORE BCUDB1 WITH RF    -
  6 record(s) selected.
```

Viewing recovery history files

Recovery history files are maintained for both the database and table space environments. These two files, `db2rhist.asc` and `db2tschg.his`, maintain a record of all transactions that would have an effect on the recovery process. The table space recovery history file assists the rollforward process in identifying only those transaction logs that must be replayed for a recovery

You can use the **db2LogsForRFwd** command to view the contents of the table space recovery history file. This allows a user to find out which log files are required for a table space rollforward operation

```
db2logsForRfwd DB2TSCHG.HIS -all
```

You can also use the **LIST HISTORY** command to query the database recovery history file for information on all recovery events. The database history file, located in the database directory, can be accessed using the LIST HISTORY command as follows:

```
list history rollforward all for bcudb1

List History File for bcudb1
Number of matching file entries = 1

 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  F  D  20100401102701      E       S0000000.LOG S0000002.LOG
----------------------------------------------------------------------------
```

An unmaintained recovery file can slow startup, because the DB2 software accesses and parses the recovery history files on startup, and eventually lead to a full disk as the DB2 software accesses and parses the recovery history files on startup. You can maintain the history file entries and associated database and transaction log backup images automatically. Set the **rec_his_retentn** parameter to the number of days you wish to retain history. Activate the process by setting the **auto_del_rec_obj** parameter to `ON`.

Note that table space backup images are not currently maintained through this parameter, but you can manage them by using the db2adutl command. You can automate the removal of table space backup images by using scripts.

The **PRUNE** command can be used to manually remove entries from both history files. The following command removes history files in the active path that were created before the S0000142.LOG file was created:

```
prune logfile prior to S0000142.log
```

Conclusion

Planning a recovery strategy is not a post-implementation task. Understanding how decisions made during the implementation cycle impact the backup and recovery objectives can help avoid problems post implementation. By clarifying the recovery objectives early in the project cycle you will be better positioned to assess the impact of design decisions and configure the system to meet the objectives. In building a flexible backup and recovery strategy that is focused on protecting your data and maximising system availability you will be better equipped to adapt to changes in your data warehouse environment as it grows and matures.

Summary of key recommendations

- Document your recovery objectives and use them as reference points when designing your backup strategy.
- Reconcile your recovery objectives with the design and configuration of your data warehouse to ensure that they are achievable.
- Identify pressure points and weaknesses in your environment because these are the most likely areas to fail, prompting a recovery.
- Avoid using PIT recovery for ETL applications because the required offline backups are inefficient.
- Deploy two tape drives per data module to increase recovery speed.
- Run frequent table space backups to minimize the number of transaction logs that are required in a recovery scenario.
- Use online backups to minimize downtime.
- Measure the success of your backup strategy by whether the backups are completed with minimal effect on resources, rather than by elapsed time.
- Focus both the backup and recovery strategies on active data. This data might account for just 20 - 25% of the overall data warehouse.
- Use tools to speed up the recovery process. You can use the **db2adutl** command to retrieve logs for rolling forward while the restore is being processed.

Appendix A. Configuration of test system

The test system that is referenced in this paper is based on an InfoSphere Balanced Warehouse D5100. The InfoSphere Balanced Warehouse (now called IBM Smart Analytics System) product offers a preconfigured data warehouse environment providing stability and performance that reflects IBM best-practices guidelines.

The test system consists of three IBM System x3650 M2 machines, one that was configured for the administration module and two that were configured for the data modules. The administration module was configured to have a single database partition containing the catalog and coordinator functions together with single-partition data warehouse tables. The data modules each comprise four database partitions that contain partitioned tables.

TSM V6.1 was installed and configured on the DB2 servers. The landing disk was configured to cater for 1.5 times the sizes of the daily logs and backups that were expected on the busiest days, Sunday and Wednesday. Two tape drives, the recommended number for the environment, were configured.

Appendix B. Configuring a DB2 instance for TSM

The following procedure shows how to configure TSM on modules where the DB2 software is installed in an environment similar to the test environment that was used for this paper.

1. Configure the TSM client on the servers where the DB2 software is running, as follows:

a. Log in to the data module as root, and navigate to where you installed the TSM client.

b. In the `/opt/tivoli/tsm/client/api/bin64` directory, copy the file, `dsm.opt.smp`, to the `dsm.opt` file, and copy the sample xxx file, `dsm.sys.smp,` to the `dsm.sys` file.

c. Add the data module as a stanza entry in the `dsm.opt` file, as shown in the following example:

```
*******************************************************************
* IBM Tivoli Storage Manager                                      *
*
* Sample Client User Options file for UNIX (dsm.opt.smp)          *
*******************************************************************
This file contains an option you can use to specify the TSM
server to contact if more than one is defined in your client
system options file (dsm.sys).  Copy dsm.opt.smp to dsm.opt.
If you enter a server name for the option below, remove the
leading asterisk (*).
*******************************************************************
* Servername       Point to beluga002 LAN or beluga002 LAN-free as required
SERVERNAME beluga002
```

d. Modify the `dsm.sys` file as shown in the following example, including adding an entry for LAN-free backups by setting the **ENABLELANFREE** parameter to YES:

```
*******************************************************************
* IBM Tivoli Storage Manager                                      *
*
* Sample Client System Options file for UNIX (dsm.sys.smp)        *
*******************************************************************
This file contains the minimum options required to get started
using TSM.  Copy dsm.sys.smp to dsm.sys.  In the dsm.sys file,
enter the appropriate values for each option listed below and
remove the leading asterisk (*) for each one.
If your client node communicates with multiple TSM servers, be
sure to add a stanza, beginning with the SERVERNAME option, for
each additional server.
*******************************************************************
SERVERNAME  beluga002 LAN
COMMMETHOD         TCPip
TCPPORT            1500
TCPSERVERADDRESS   xxx.xxx.xxx.xxx
PASSWORDACCESS     generate
ENABLELANFREE NO

SERVERNAME  beluga002_LAN-free
COMMMETHOD         TCPip
TCPPORT            1500
TCPSERVERADDRESS   xxx.xxx.xxx.xxx
PASSWORDACCESS     generate
ENABLELANFREE YES
```

2. Set up the instance environment, as follows:

a. Log in to each instance as the instance owner,

b. Add the following entries to the `~/.kshrc` file in the home directory (for 64-bit DB2, replace `bin` with `bin64`):

```
export DSMI_DIR=/opt/tivoli/tsm/client/api/bin64
export DSMI_CONFIG=/opt/tivoli/tsm/client/api/bin64/dsm.opt
export DSMI_LOG=$HOME/sqllib/db2dump
```

c. In the `~/sqllib/` directory, create a file called `userprofile` that contains the three entries that you added to the `~/.kshrc` file.

3. Test TSM server and TSM client connectivity, as follows:

a. Log in to each data module as root, and run the following **export** commands:

```
export DSMI_DIR=/opt/tivoli/tsm/client/api/bin64
export DSMI_CONFIG=/opt/tivoli/tsm/client/api/bin64/dsm.opt
export DSMI_LOG=homeFileSystem of instanceID/sqllib/db2dump
```

b. From `~`*`instanceID`*`/sqllib/adsm` directory, issue the following command:

```
./dsmapipw
```

c. When prompted for a password, enter the appropriate password.

d. From the `~`*`instanceID`*`/sqllib/db2dump` directory, issue the following command:

```
chown instanceID:instanceGroup  dsierror.log
```

TSM is now configured and ready to be used for backup and recovery commands.

Appendix C. Overview of DB2 utilities

The restore, rollforward and recover utilities are used to perform data recovery.

The **RESTORE DATABASE** command restores a database or table space from the contents of a backup image. The **ROLLFORWARD** command reapplies transactions that were committed after a backup was made or while the backup process of an online backup was executing. The **RECOVER** utility performs the necessary restore and rollforward operations to recover a database to a specified time based on information found in the recovery history file.

Refer to the DB2 user guides for further reading.

Restore utility

You can use a full database backup image to recover a version of the database, as shown in the following example:

```
restore database bcudb1 use TSM taken at 20100209134904
```

Use the **REBUILD** parameter of the **RESTORE DATABASE** command to rebuild part or all of an entire database by using only table space backups and logs (alternatively, you can use full and incremental backup images). The recovery history files are referenced to determine the location of the backups and logs; these backups and logs must be in the expected location for the rebuild to succeed.

You can also recover an entire database by using the rebuild feature to restore each table space and then roll forward to the ends of the logs. As you restore and roll forward each table space, it becomes available for use. If a backup for a table space is unavailable, the associated table space is placed in the drop pending state, from which you can drop or restore the table space later.

SYSADM, SYSCTRL, or SYSMAINT authority is needed to restore into an existing database, and SYSADM or SYSCTRL authority is required to restore to a new database.

Roll forward utility

Rollforward recovery builds on a version recovery by using log files to recover a database or table space to a specific point in time. In an online backup and recovery strategy this utility will be used to roll forward to the end of the logs included in the backup image.

The following command would be issued after the restore command in the previous section:

```
rollforward database bcudb1 to end of logs and complete
```

Recover utility

The **RECOVER DATABASE** command combines the effects of the **RESTORE DATABASE** and **ROLLFORWARD DATABASE** commands by using the history files to identify which backup images and logs to use to complete a recovery. This process makes the **RECOVER DATABASE** command better suited to a database partition, data module, or full database recovery.

You can replace the sample commands that were shown for the rollforward recovery with the following command:

```
recover database bcudb1 to end of logs
```

A crash recovery is an abnormal termination of a DB2 instance due to a power failure or other unexpected event. In a crash recovery, incomplete transactions are committed if possible; otherwise, they are rolled back. If a crash occurs and the **AUTOSTART** configuration parameter is set to ON, the first database connection attempt triggers the crash recovery.
Recommended best practice is to set **AUTOSTART** to OFF so that you can assess and manage the recovery process as per your recovery objectives.

Further reading

- "Developing a backup and recovery strategy" topic

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.ha.doc/doc/c0005945.html

- "Database Physical Design" DB2 best-practices paper

  http://www.ibm.com/developerworks/data/bestpractices/databasedesign/

- "Unleashing the power of table partitioning in your DB2 Warehouse" article

  http://www.ibm.com/developerworks/data/library/techarticle/dm-1006tablepartitioning/index.html

- Tape systems Web page (contains information about LTO tape drives)

  http://www-03.ibm.com/systems/storage/tape/

- "Compatibility of online backup and other utilities" topic

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0021554.html

- "LAN-free DB2 Backups to TSM Recommendation" technote

  http://www-01.ibm.com/support/docview.wss?uid=swg21287060

- "System commands" topic

  http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?nav=/3_6_2_4

- Tivoli Storage Manager (TSM) overview

  http://publib.boulder.ibm.com/infocenter/tsminfo/v6/index.jsp?topic=/com.ibm.itsm.srv.doc/c_tsmintro.html/

Contributors

John Bell

IBM Senior Data Warehouse Architect

Sermsak Sukjirawat

WW Smart Analytics System Practices, IM Software Services

Lydia Siu

DB2 Data Warehouse QA Team

Ilker Ender

DB2 Data Warehouse QA Team

Joyce Coleman

IBM Smart Analytics System - Development

Katherine Kurtz

IBM Smart Analytics System Best Practices Manager

Simon Woodcock

Infosphere Warehouse Technical Sales Consultant

Nela Krawez

IBM Smart Analytics System Solutions Development

Haider Rizvi

IBM Smart Analytics System Architect

Notices

Trademarks