A Practical Guide to Backup and Recovery of IBM DB2 for Linux, UNIX and Windows in SAP Environments

Part 1 – Backup and Recovery – Overview

Version 1.4

IBM SAP DB2
Center of Excellence

Revision date: **20.08.2009**

Authors: **Olaf Depper**
Edgardo G. Koenig
Hans-Jürgen Moldowan
Thomas Rech

About this paper

The database is the foundation of the SAP® system. All relevant business data and also the application code that is executed by the SAP application server are stored in the database. It is critical to have an effective and secure strategy for database backups and recovery planned, implemented and tested. Therefore a detailed understanding of the backup and recovery procedures for the database of the SAP system is required.

The following document is part one out of a series of documents that will cover the essential topics around backup and recovery for IBM® DB2® for Linux®, UNIX®, and Windows® (DB2 for Linux, UNIX and Windows) together with SAP Applications based on SAP NetWeaver®.

To manage the balancing act between quick start and advanced topics, the series of papers is planned to describe with the following dedicated topics:

- Backup and recovery overview (this paper)
- Backup integration with vendor solutions
- Backup tuning and optimizations
- Backup best practices
- Backup and recovery in partitioned environments (DB2 Database Partitioning Feature)

It is planned to publish the documents sequentially. For updates and new editions of these documents, check the appropriate information sources, for example, IBM developerWorks® or SAP Developer Network.

The following paper introduces the basics of backup and recovery for DB2 database. It explains the architecture for backup and recovery, the most relevant commands and discusses the DB2 log file management. In addition, the integration of the DB2 backup and recovery architecture into SAP NetWeaver™ is described.

The document is designed as an introduction. It is intended to provide a fast start for the development and implementation of a reliable backup and recovery strategy.

Table of Contents

1 Backup and Recovery – Overview

The following chapter highlights some of the architectural components with respect to the backup and recovery architecture. It explains the DB2 objects relevant for backups and the different methods of backup, for example, offline backup or online backup. The chapter describes the DB2 backup architecture and the DB2 logging. This information is aimed to enable you to develop and implement a DB2 backup and recovery strategy.

1.1 Database Objects

DB2 databases are created in DB2 instances on the database server. A DB2 instance is also called "database manager" as it provides the environment to host one or more databases. On one physical server, multiple DB2 instances can be created. In an SAP environment each instance holds only one database. While the database name or alias is determined by the SAP system ID (<SAPSID>) itself, the DB2 instance is called db2<dbsid>.

Each database has multiple table spaces and a dedicated set of log files. You can mirror DB2 online log files using database means. Each table space consists of one or more containers. Containers can be directories, files or raw devices. The table space containers hold the database objects like tables and indexes. Table spaces are a logical element with the table space containers as the physical representation on disk. In a typical DB2 installation, the majority of the disk space is used by the table space containers and log files. The DB2 instance and the database representation itself consist mainly of configuration and administrative files in the respective directories.

The instance directory contains, for example, the database manager configuration files, the system database directory, the node directory and the instance registry. It also holds the links to the installed database software as well as some executables that are local to the instance.

The database directory contains the critical configuration files of the database. These files define, for example, the buffer pools, table spaces, or the configuration of the database. The database directory is also the location for the log file header as well as the history file.

In addition to the instance and database configuration, the DB2 system has a profile registry. Settings in the DB2 profile registry can affect the instance (for example, parameters that are related to communication) or the database (for example, parameters that are related to the behavior of the SQL optimizer). In an SAP environment, one of these DB2 registry variables is 'DB2_WORKLOAD' which in our case has the value 'SAP'. Most of the parameters are set in the instance profile registry that is stored in the instance directory. Some entries, however, are set for the database server. These registry settings are set in the global registry which is stored in the file *'/var/db2/global.reg'* in Linux, UNIX environments. In Windows environments, the DB2 profile registry is stored in the Windows registry (in the hive with key: HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\GLOBAL_PROFILE).

The following Figure 1-1 shows the simplified relationship between the different objects discussed.

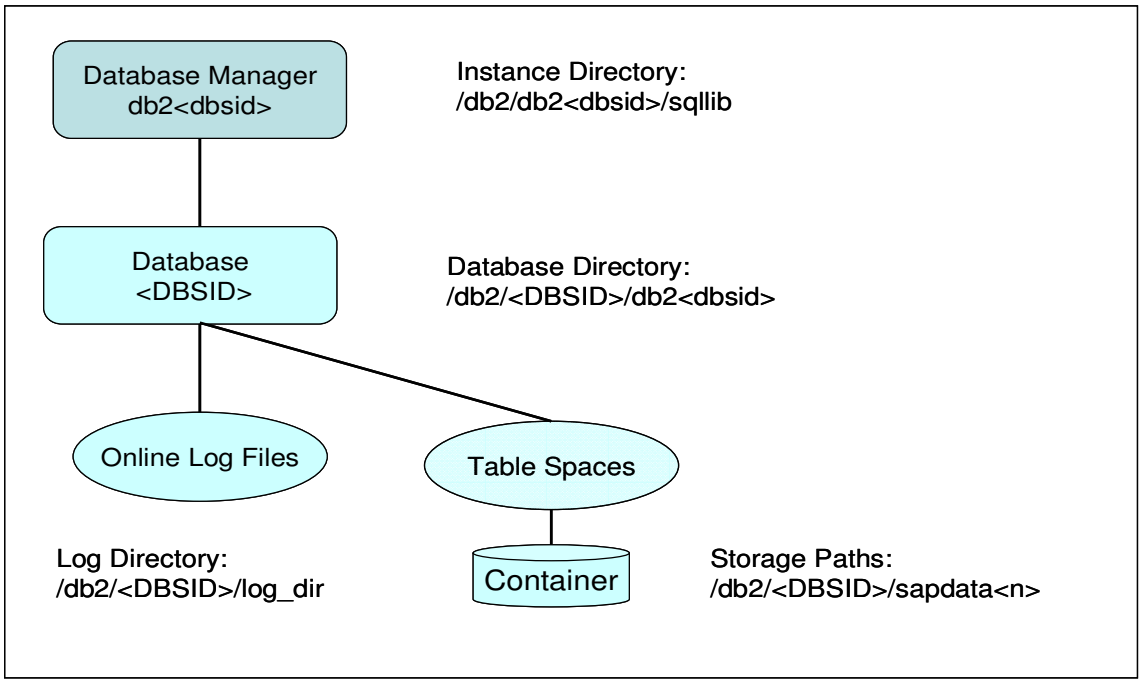


Figure 1-1: Database Objects

In addition to the already mentioned objects, the DB2 database system has further level of logical objects or group of objects – the database partitions and database partition groups. Another and older term for database partition is node. These logical objects are related to the database partitioning feature (DPF) of DB2 for Linux, UNIX and Windows. DPF-related topics are not described in this document. Backup and recovery in partitioned database environments will be described in an upcoming document of this series. For non-partitioned environments, the default partition (node) always is '0000'.

Why are these objects important to understand in the context of the DB2 backup utility? The DB2 database backup saves all information that is necessary to recover the database. However, as described, objects exist outside the database. Consequently, they are not part of the DB2 database backup. The data itself, the database and table space configuration and – as the default behavior of DB2 9.5 and higher – the log files are saved as part of an online database backup. The database manager configuration and the registry, however, are not included in this image. Although these configuration files are not essential for a full disaster recovery, it is of some interest to understand this fact. For more information about how to save and recover these configuration files, see Chapter 4.1 "DB2CFEXP / DB2CFIMP".

1.2 Logging with DB2 databases

The DB2 database system uses a write-ahead logging algorithm as the transaction recovery method that supports fine-granularity logging and partial rollbacks. Write-ahead logging forces the log record to be written to the log file for an update before the corresponding data page is written to disk. All log records for a transaction must be written to the log files before a COMMIT is completed. The database stores information that is related to the changes in a set of log files in the database log directory.

The default location of the DB2 log files in SAP environments is '/db2/<DBSID>/log_dir'. The DB2 database management system automatically appends the number of the database partition (in the form 'NODE<nnnn>') to the log path to maintain uniqueness for DPF environments. This node number is appended also in non-partitioned environments. For example, in a non-partitioned DB2/SAP system the path for the online log files would be '/db2/<DBSID>/log_dir/NODE0000'.

The DB2 database system distinguishes between primary and secondary log files. Both, primary and secondary log files reside in the database log directory. During startup, the database pre-allocates all primary log files in case they do not already exist. In case that long-running transactions fill up all

primary log files, the database engine allocates secondary log files one at a time as needed. The number of primary and secondary log files as well as the size of the log files is defined in the database configuration.

Database log files are crucial for data recovery. The DB2 system knows three types of data recovery:

- Crash recovery (also known as restart recovery)
  Crash recovery brings back the database to a consistent and useable state after an unexpected shutdown (for example, after a power failure).
- Version recovery
  The restoration of a previous version of the database, using an backup image.
- Rollforward recovery
  Transactions recorded in database logs are applied following the database restore operation. This method allows the recovery of the database (or table space) to a particular point in time.

Crash recovery and version recovery are supported by every DB2 database. Whether rollforward recovery is supported depends on the logging mechanism used by the database.

Circular logging is the default logging mechanism of a newly created DB2 database. Only full offline database backups are allowed with this logging method. The DB2 database handles log files in a circular fashion: If the last log file is full, the database begins writing to the first log file again. Databases using circular logging do not allow online backup or rollforward recovery. Circular logging is **not** supported for production databases in SAP environments.

The recommended logging mechanism for production environments is archival logging. If archival logging is used, a log file is archived to an archival location as soon as the log file is full (that is no more log records can be written to the log file). A database that uses archival logging can be backed up online. To reach a specified point in time, you can perform a rollforward recovery. A database that uses archival logging is therefore also called “recoverable”.

The following Figure 1-2 provides an overview of circular and archival logging:

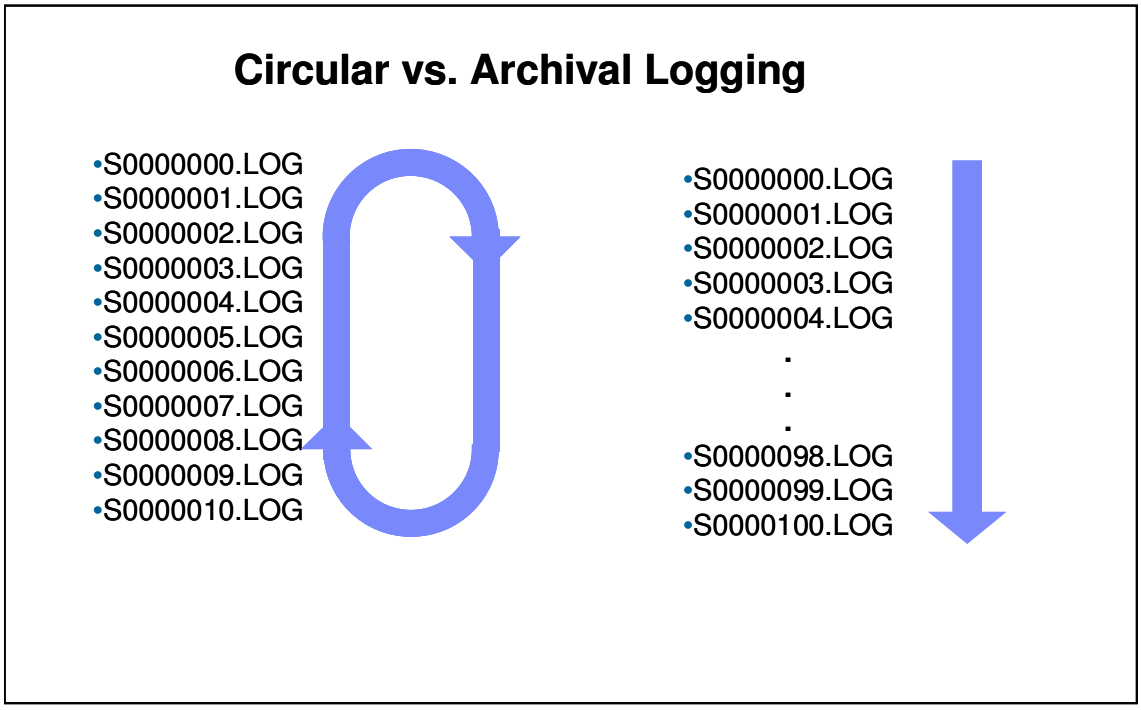


Figure 1-2: Logging Methods for DB2 databases

You have to explicitly activate archival logging in the database configuration. Whether a DB2 database is using archival or circular logging is defined by the log archive method database configuration parameter (LOGARCHMETH1 and LOGARCHMETH2). If both parameters LOGARCHMETH1 and LOGARCHMETH2 are set to OFF, the database uses circular logging. You enable archival logging by

setting either both or one of the configuration parameters LOGARCHMETH1 and LOGARCHMETH2 to a value other than OFF.

For a recoverable database, the DB2 engine archives a log file as soon as it is full or explicitly archived. However, this log file might still contain information that could be required for crash recovery. Therefore, the log file stays in the log directory as long as it is required for a crash recovery. From a DB2 database point of view, these log files are considered to be 'active'.

Once a log file has been archived successfully by the DB2 log manager and it does not contain any open transactions (and is therefore not required any more for a crash recovery), the log file is considered to be 'archived'. Archived log files are usually not deleted by the database. Instead, when a new log file is required, the DB2 database will rename the oldest archived log file and use it again. The DB2 database will rename a log file instead of allocating a new file to reduce I/O operations. Archived logs are not renamed immediately once they have been copied to the archive. For that reason it is common to see old log files in the log directory when no new active log files are required.

Note: You should never move log files from the log directory manually while the database is online. The DB2 system automatically allocates and removes of log files. If you delete an active log file, the DB2 instance shuts down the database. The database cannot be restarted anymore and has to be restored.

The following figure provides an (incomplete) overview of the DB2 logging mechanism with the log manager (db2logmgr) as the core database engine component.

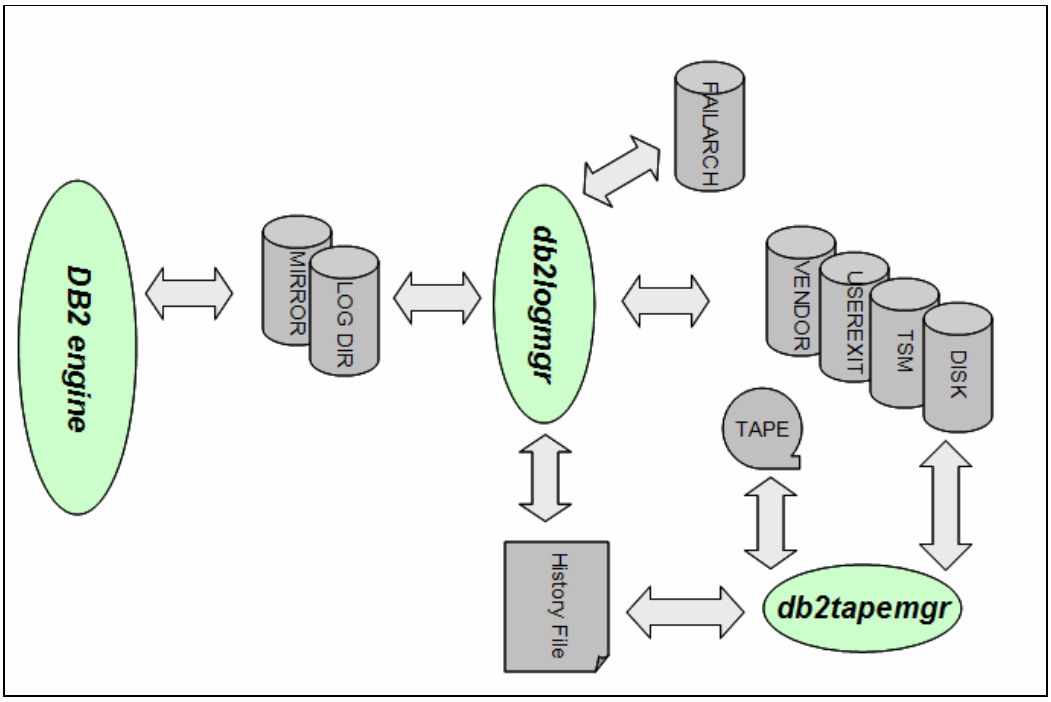


Figure 1-3: The DB2 Log Manager

The DB2 engine writes and reads from the log files located in the log directory of the database. Once a log file is full, the DB2 log manager process (db2logmgr) controls the archive process of the log file to the different possible locations. In case that a tape drive is defined as the archive target location, an additional DB2 process, the DB2 tape manager (db2tapemgr), is spawned. The DB2 log manager stores information about all archive operations in the history file of the database. It is possible to define a fail over archive location that is used in case the standard archive location is temporarily unavailable. If the database engine requires access to an archived log file, for example, as part of a recovery, the DB2 log manager retrieves the log files from the archive location.

For more details on how to configure the DB2 logging, see chapter 2 "Log File Management – Details".

1.3 Backup Architecture

The DB2 backup utility reads the content of table spaces and writes this data to the backup media. The utility does not copy complete table spaces or, to be more precise, the table space container files to the backup media. Instead, the backup utility saves the content of table spaces. In addition to the table space data, a DB2 backup automatically saves all the configuration files that are located in the database directory as part of the backup image. The log files that are required to recover the database to the earliest consistent point can also be included in the backup image. By default, log files are included in online backups as of DB2 Version 9.5 and higher.

A DB2 backup command, for example, '**`BACKUP DB WGN ONLINE USE TSM`**' reads the pages out of the table spaces and sends them to the defined target, in our example IBM Tivoli® Storage Manager. The command also saves the database configuration files and the log files that are required to recover the database.

Besides Tivoli Storage Manager, the DB2 database system supports backups to disk and tape. In addition, the DB2 backup utility provides interfaces that are used by vendor backup solutions like EMC NetWorker® or HP OpenView Storage Data Protector®. For example, the DB2 command 'BACKUP DB WGN LOAD <vendor library>' reads the pages out of the table spaces and sends them to the vendor library together with the database configuration files. Since in this case the backup is an offline backup, no log files are included in the backup image.

A special option of the DB2 BACKUP command is to use the 'SNAPSHOT' keyword. In this case, the DB2 backup utility uses storage capabilities to create a backup image on disk level. The default behavior for a snapshot backup is an offline database backup of all database paths, for example, all table space containers, the database path, and the log paths. Although this is an offline backup, the log files are included in this case. Unlike the other backup target options, this offline backup does not read the data out of the table spaces. Instead disk images are created.

Since a snapshot backup differs somehow from the 'classic' backup to tape, disk or another backup solution it will not be covered in this paper. For more information about snapshot backups in an SAP environment, see the Whitepaper "*Setup and Configuration of the DB2 Snapshot Backup with IBM N-Series Storage*" available in the SAP Developer Network at:

http://sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/0019590b-658d-2b10-24bd-9c6882b2b009

Getting back to the type of backups covered in this paper, some more details about the process model of a backup and its main influencing factors need to be described. You can call the backup command with several parameters that specify, for example, the number and size of buffers, the degree of parallelism, the number of sessions to open and whether the backup image should be compressed or not. These parameters are described in chapter 3 "Backup and Recovery – Details". For now we concentrate on some parameters that are important to understand the backup concept.

The following picture describes the data flow and the participating components when you perform a DB2 backup.

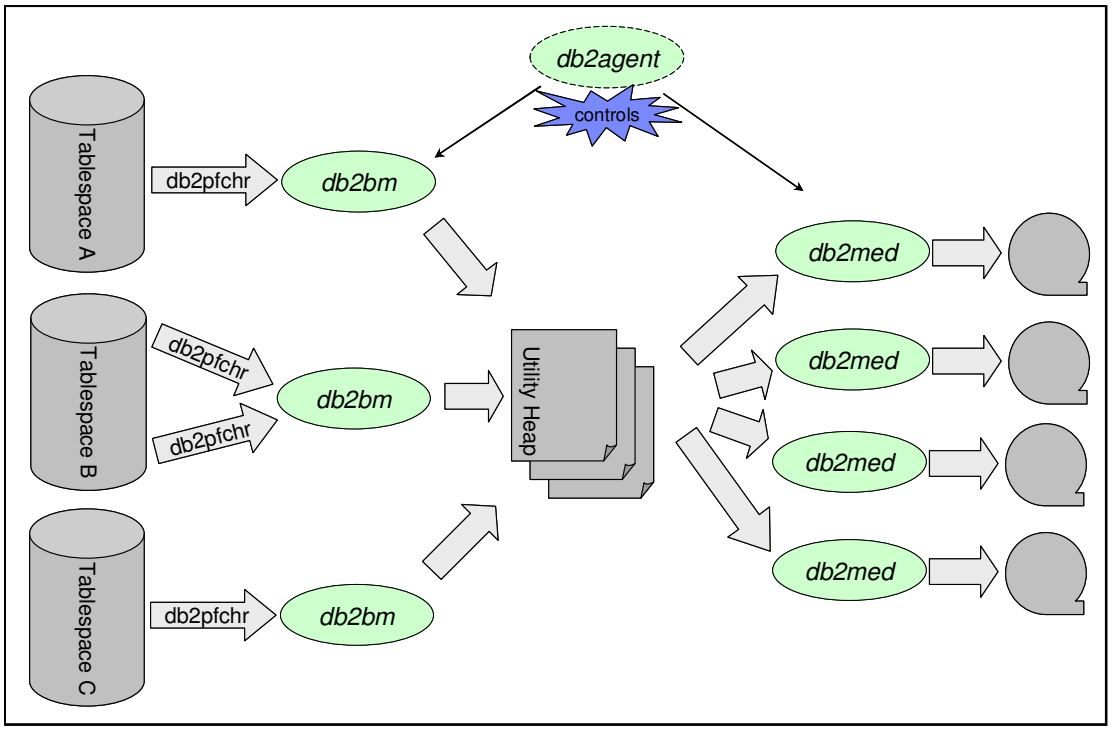


Figure 1-4: The DB2 Backup Components

If a DB2 backup command is issued, an engine dispatchable unit (EDU) called agent (db2agent) is spawned. This db2agent controls the backup session. The DB2 engine starts additional EDUs called buffer manipulators (db2bm). The buffer manipulators are working together with the prefetcher processes (db2pfchr) to retrieve the data from the table spaces into the backup memory area. This memory is part of the Utility Heap defined in the database configuration. Only one buffer manipulator is assigned to a table space while multiple prefetchers can read the data into the memory for a single table space. The data retrieved is then transferred from the backup memory to the media controller processes (db2med). These EDUs are responsible for moving data from the backup buffers to the target media device(s).

The number of prefetchers depends on the database configuration and is determined by the parameter 'NUM_IOSERVERS'. Backup buffers, the number of DB2 buffer manipulators, and the number of media controllers are defined in the BACKUP command itself with the following keywords:

- WITH <number> BUFFERS
  Controls the number of backup buffers
- BUFFER <size>
  Determines the size of one backup buffer
- PARALLELISM
  Sets the number of table spaces which can be read in parallel by the backup utility (influences the number of db2bm EDUs)
- OPEN <number> SESSION or <number of target devices specified>
  Determines the number of DB2 Media Controllers (db2med)

The backup parameters mentioned so far (parallelism, sessions, and buffers) are optional. If not specified manually, the DB2 backup utility automatically tunes these parameters based on the number of table spaces in the database, the available memory and the number of CPUs of the database server.

DB2 offers the functionality to throttle database utilities, for example, BACKUP or RUNSTATS. This allows regulating the impact that a running utility might have on the system performance. Throttling might make it easier to start a database backup in parallel with high workloads. Using backup throttling, the media controllers slow down and by doing so, the read I/O decreases. As a consequence, the backup itself will most probably take longer.

There is one important design aspect to ensure optimal backup performance that is independent from the IT infrastructure or the backup media used: the table space layout of the database.

Looking at SAP databases, the data is usually not evenly distributed over the table spaces. Some of them, for example, the <SAPSID>#BTABD and <SAPSID>#BTABI table spaces hold the majority of data. On the other hand many table spaces like the <SAPSID>#USER1D table spaces only hold few data. The following figure shows an example list of the largest table spaces in an SAP database. In this case, the FCK#BTABD table space is by factors larger than the next smaller table space.

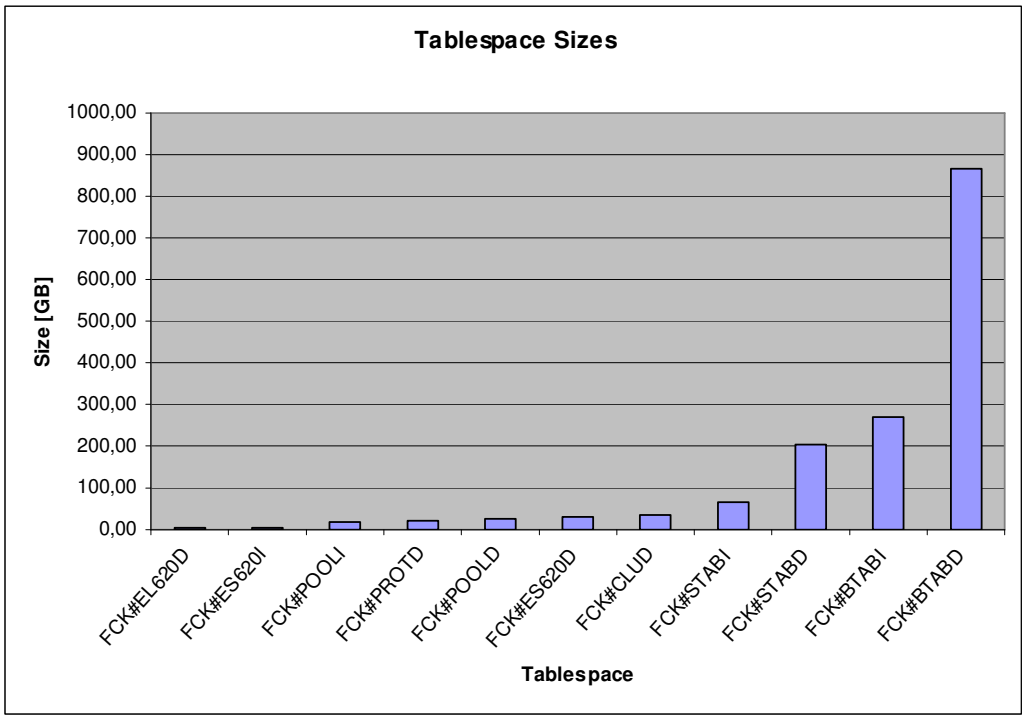


Figure 1-5 Table Space Layout (Not Optimized)

As described earlier, the PARALLELISM parameter of the DB2 backup command influences the number of table spaces which can be read in parallel by the backup. On process or thread level, the PARALLELISM parameter defines the number of DB2 buffer manipulators (db2bm) that are used for the backup. Each db2bm EDU is assigned to a specific table space. Using a value for the PARALLELISM parameter that is larger than the number of table spaces being backed up will therefore not increase the backup throughput.

The best way to ensure optimal parallelism of a database backup is therefore to distribute the data as even as possible over the table spaces, for example, by using dedicated table spaces for the largest tables. The following figure shows such an optimized layout:

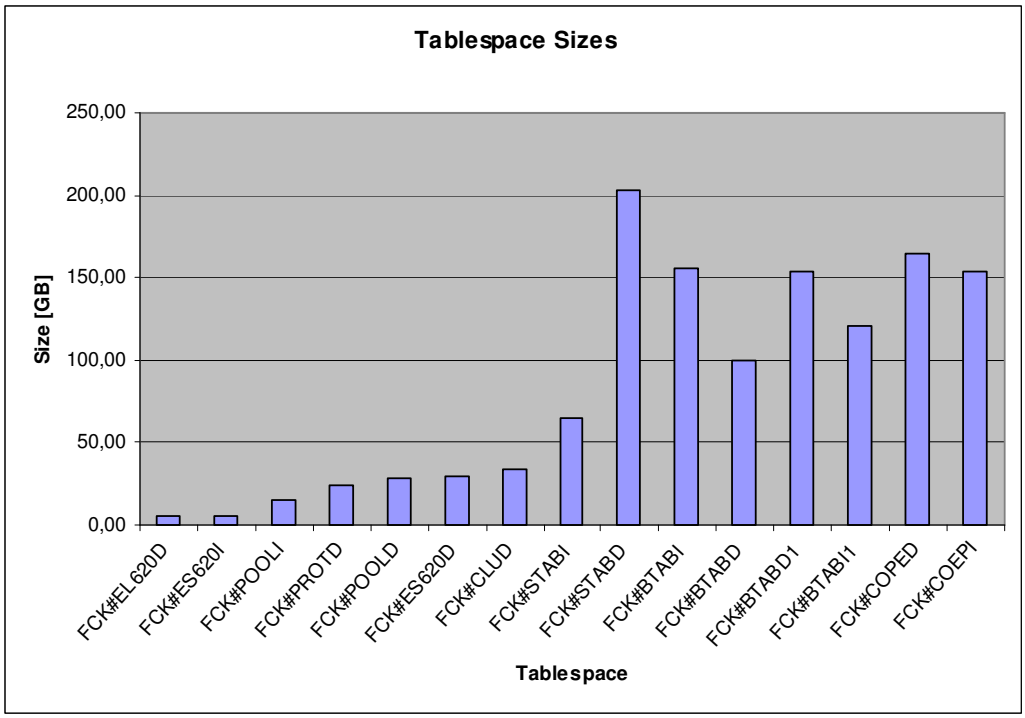


Figure 1-6 Table Space Sizes (Optimized)

Figure 1-6 shows a much better distribution of data to table spaces: Instead of 80 percent residing in one big table space (as shown in Figure 1-5) the data has now been distributed over seven table spaces with similar sizes. This type of setup supports optimal backup parallelism.

Beside the backup aspect of such a design, moving large tables to their dedicated table spaces is a common best practice for large SAP databases. There are several SAP tools available for DB2 for Linux, UNIX and Windows, for example, online table move or the DB6CONV tool. These tools allow you to move a table online to a new table space without significant impact on the production database.

The DB2 backup utility also requires some knowledge of a concept called table space high-water mark (HWM). The high-water mark describes the highest page which contains data in the table space. Necessarily not all pages up to this high-water mark are filled with data. Typically, due to deletes, reorganization of tables or implementation of row compression, the HWM might be a lot higher than the number of used pages in a table space. The following Figure 1-7 illustrates this fact.

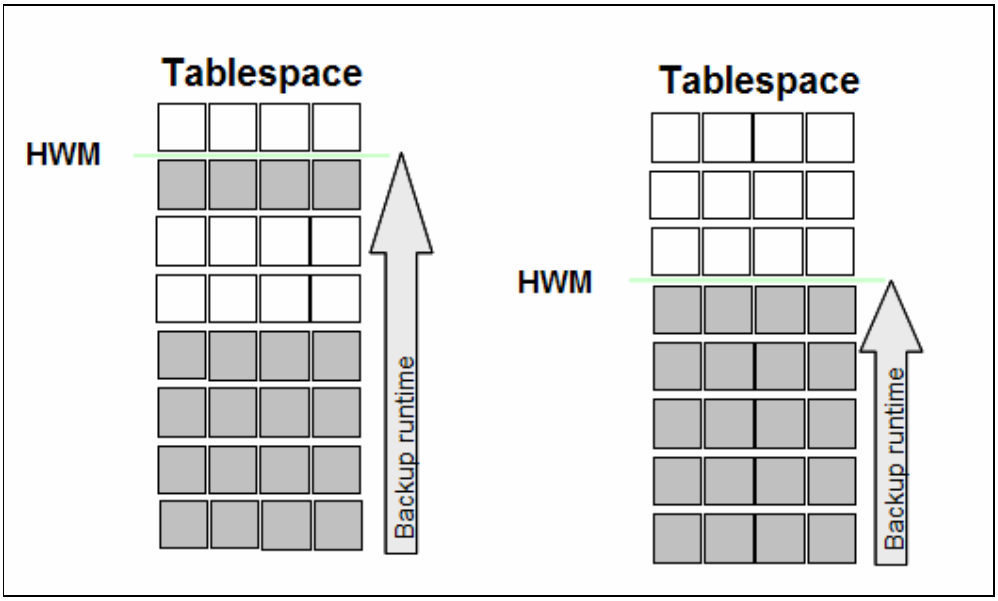


Figure 1-7: Table Space High-water Mark

In the figure above, both table spaces contain the same amount of used pages. However, for the table space on the left side the HWM is higher. As mentioned earlier, the DB2 backup utility does not create a copy of the table space container files, but saves the content of the table spaces instead. The backup utility reads all pages inside a table space up to the HWM. Therefore a backup can take longer on the table space having a higher HWM than compared to the table space illustrated on the right with a lower HWM.

1.4 Integration of the DB2 Backup Utility in SAP Environments

The key concept of the autonomic features is helping system administrators by automating regular maintenance tasks. The integration of the DB2 administrative tasks into the SAP system is following the same principle. SAP delivers a graphical user interface, that is, the DBA COCKPIT to manage DB2 database backups. The DBA Cockpit contains a lot of functionality including backup scheduling (which is part of the DBA Planning Calendar in the DBA Cockpit).

Since the DB2 backup utility and the log manager are integrated parts of the database engine, an API is available to call the required functionality from within the SAP system. So probably an administrator uses the DB2 commands only to set up the backup and log archiving function. Once the system is up and running, you can use the DBA Cockpit (SAP transaction DBACOCKPIT) to manage all the backups. For more information about the integration of the DB2 BACKUP command into the DBA cockpit see chapter 5 “DB2 and SAP Integration”.

2 Log File Management – Details

In this section we explain the concept of log file management and its importance for the proper definition of a backup and recovery strategy. We will describe the basic database configuration parameters to configure the log file management for the DB2 database and also explain how you configure the different logging types that are available.

2.1 Basic Log File Management Concepts

The DB2 engine logs all changes that made to database objects in the database log files. Each database log file contains a number of log records. These log records store the information about the different changes that were performed by the transactions running against the database. In the case of a recovery, the information stored in the log files is used by the database engine to bring back the database to a consistent state. Therefore, log files are a critical element in a database and you must handle them with special attention.

One of the most important considerations when you set up a database is the location of the log files. By default, log files are located in the SQLOGDIR directory that is created in the database directory after the execution of the CREATE DATABASE command. As mentioned earlier, in SAP environments this location is changed during the SAP system installation. The SAP standard location for log files is */db2/<DBSID>/log_dir*.

You can change the log path by setting the database configuration parameter NEWLOGPATH as shown in the following example:

```
UPDATE DB CFG FOR SGE USING NEWLOGPATH /db2/SGE/log_dir1
```

The execution of this command does not initiate an immediate relocation of the log files. The new log path is used the next time the database is restarted. The DB2 database then starts to create the log files in the new directory.

For high availability environments the mirror logging functionality of DB2 provides additional protection against disk failure or human errors. To enable mirror logging, DB2 for Linux, UNIX and Windows provides the MIRRORLOGPATH database configuration parameter. By setting this parameter, the DB2 database automatically writes log files in both the active and the mirror log path. For example, to activate mirror logging for database SGE, we used the following command:

```
UPDATE DB CFG FOR SGE USING MIRRORLOGPATH /db2/SGE/log_dir2
```

An important element in the configuration of the log file management is the number of primary and secondary log files. The LOGPRIMARY database configuration parameter indicates the number of primary log files. The LOGSECOND database configuration parameter determines the number of secondary log files. Both primary and secondary log files are files that reside in the database log directory. The difference between each of them is the moment when they are allocated. While primary log files are allocated (in case they do not already exist) when the database is activated, secondary log files are only allocated if they are required to store information of a transaction that is running a long time.

The DB2 log files start with the name S0000000.LOG. The number is automatically incremented every time a new log is created. The size of the log files can be modified by setting the database configuration parameter LOGFILSIZ. The log file size is given in 4k pages.

To change the values of the parameters, execute the following commands:

```
UPDATE DB CFG FOR <db> USING LOGPRIMARY <value>
UPDATE DB CFG FOR <db> USING LOGSECOND <value>
UPDATE DB CFG FOR <db> USING LOGFILSIZ <value>
```

To activate the changes to both the LOGPRIMARY and LOGFILSIZ parameter, you have to restart the database. Note that the DB2 database management system creates additional (when changing number of primary logs) or new log files with the new size (when changing log file size) in the log directory during the database activation. This process might take some minutes depending on the I/O performance of the log directory. You can change the LOGSECOND parameter online.

The amount of log space that is available for the database engine is determined by the number and size of primary and secondary log files ((LOGPRIMARY + LOGSECOND) x LOGFILSIZ). To avoid log file full situations (SQL error SQL0964C), it is important to allocate sufficient log space for a database. Another consideration is the available disk space – the configured log files (both primary and secondary) must fit into the specified directory.

You can define "infinite log space" by setting the database configuration parameter LOGSECOND to a value of "-1". As soon as a secondary log file is full, the DB2 database archives the log file and creates the next secondary log file. This way, the log disk only holds the primary log files including one secondary log file. However, use "infinite logging" with care and only for special workloads: Every rollback or crash recovery might need to recover log files from the archiving media and therefore slowing down the process.

The following table summarizes the possible values for each of these parameters (DB2 9.5 Fix Pack 3):

Parameter	Value range	Consideration
LOGPRIMARY	2-256	If LOGSECOND is not equal to -1, LOGPRIMARY + LOGSECOND can not exceed the value 256. If LOGSECOND is equal to -1, the maximum value for LOGPRIMARY is 256
LOGSECOND	-1, 0-254	
LOGFILSIZ	4-1048572	Number of 4 KB pages

Table 1: Possible Values for Number and Size of Log Files

The upper limit of log file size (1048572 4 KB pages), combined with the upper limit of 256 for the number of log files (LOGPRIMARY + LOGSECOND), gives an upper limit of 1024 GB of active log space in DB2 9.5 (Fix Pack 3 and higher).

To minimize I/O operations, the changes to the database are first written to a log buffer. The LOGBUFSZ database configuration parameter defines the size of the log buffer. Once the transaction is committed or the log buffer is full, the log buffer content is written to the corresponding log file on disk.

2.2 Archiving of Log Files

As described in the introduction of this document DB2 for Linux, UNIX and Windows has two different logging methods: circular and archival logging. Circular logging is the default logging type after the creation of a database. Because of several restrictions that apply if circular logging is used, this method is not supported for production systems in SAP environments. Instead, you have to configure the database to use archival logging.

Whether the database is using archival or circular logging is determined by how the database configuration parameter LOGARCHMETH1 is set. If you set this parameter to a value other than OFF the database uses archival logging and is considered to be "recoverable". To update the configuration, you can use the following command:

```
UPDATE DB CFG FOR <db> USING LOGARCHMETH1 <value>
```

If you set LOGARCHMETH1 from OFF to any other value, you have to restart the database to activate the change of the archiving method. The database will be put in "Backup pending state" after the log files have been formatted. Connects to the database are only possible after a full offline backup of the database has been performed.

Depending on the archiving method and the target destination for archived log files used, you have to set the LOGARCHMETH1 database configuration parameter accordingly.

The following table summarizes the possible values:

Value	Pattern / Description
OFF	Archival logging is not being used
DISK	DISK:<path>: Disk location to where log files are archived
TSM	TSM[:MgmtClass] If no management class is provided, the default in the local Tivoli Storage Manager server is used; otherwise, the Tivoli Storage Manager management class used as target destination is provided.
VENDOR	VENDOR:<library name> Name of the vendor library being used to archive logs
LOGRETAIN	LOGRETAIN Archived log files will remain in the log directory and are not reused by the DB2 database. This enables the database for rollforward recovery but does not archive the log files.
USEREXIT	USEREXIT The DB2 engine will call a user provided db2uext2 executable to archive log files.

Table 2: Possible Values for LOGARCHMETH1

In recoverable databases an EDU called db2logmgr (DB2 Log File Manager) is used to handle archived log files. In addition to LOGARCHMETH1, the DB2 system offers several database configuration parameters to configure further options that are related to archival logging. For example, you can enable a secondary archive destination for archived log files or define the behavior of the DB2 log manager in case an archive of a log file fails.

In the next sections, we introduce the most important additional parameters related to archiving of log files. You can find a complete list of parameters related to logging in the IBM DB2 Information Center.

The LOGARCHMETH2 database configuration parameter is used in a similar fashion as the LOGARCHMETH1 database configuration parameter. It lets you define a secondary location to archive log files. You cannot use this parameter when LOGARCHMETH1 is set to LOGRETAIN or USEREXIT.

The log archiving methods defined in the LOGARCHMETH1 or LOGARCHMETH2 configuration parameters might require additional options. These options can be set in the database configuration parameters LOGARCHOPT1 (configures log archive method 1) and LOGARCHOPT2 (configures log archive method 2).

The ARCHRETRYDELAY database configuration parameter specifies the amount of time to wait, in case an attempt to archive a log file does not finish successfully. After the specified number of seconds, it is attempted to archive the log file again. The NUMARCHRETRY database configuration parameter specifies the number of times to attempt to archive a log file.

Once the defined number of failed log file archival attempts is reached, the log file is moved to the directory specified in the FAILARCHPATH database configuration parameter. The directory defined in this database configuration parameter is used as temporary storage for the archived log files. This can be used to avoid the log directory to run out of disk space.

Note: For backward compatibility, DB2 for Linux, UNIX and Windows still supports the old database parameters LOGRETAIN and USEREXIT. However, these two parameters are deprecated and should not be used any more. Updating LOGARCHMETH1 to values USEREXIT or LOGRETAIN automatically updates the database parameters USEREXIT and LOGRETAIN.

2.3 Log File Chaining

Every DB2 log file is associated with a log chain. A log chain is defined as the collection of log files archived during a particular period of time. Every time a roll forward operation to point in time or a restore without roll forward is executed, the DB2 engine automatically starts a new log chain.

When starting a new log chain, the DB2 engine truncates the last applied log file and continues to use log files within a new log chain. Creating a new log chain has no influence on the naming of the online log files and is transparent to the user. However, the log chain has some effect on the archived logs. If the archiving method is set to disk, the log files are stored in a hierarchy of subdirectories under the path specified for the log file.

If the archiving method is set to DISK:`<log_archive>`, the hierarchy looks as follows:

`<log_archive>/<instance>/<database>/NODExxxx/Cyyyyyyy/Szzzzzzz.LOG`

The subdirectory "*Cyyyyyyy*" represents the log chain. The naming starts with "C0000000" and is incremented every time a new log chain starts.

This ensures that no archived log files are overwritten by a new log chain. A situation where log files with identical names but different log chain exist can occur, if a database is recovered to an earlier point in time. The following section describes this situation in more detail.

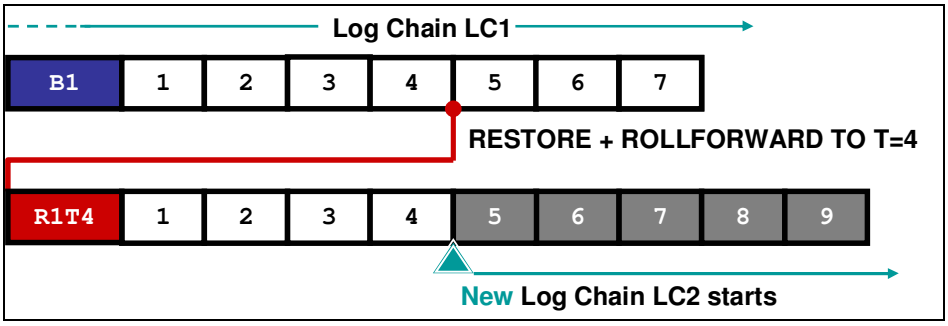


Figure 2-1: DB2 Log Chains

As shown in Figure 2-1, a new log chain is started when a restore and roll forward to point in time is performed. After taking backup B1, the DB2 system stores transactional information in the log files 1 to 7. The database will continue to write new log files as usual. However, the execution of a restore using backup image 1 and a roll forward of the database to point in time T4 implies the start of a new log chain. Therefore, log files 5 to 7 are overwritten with new transactional information. In this situation, the database has one version of log files 5 to 7 that belong to log chain LC1 and another version of these logs that belong to the newly started log chain LC2 in its log archive target.

Log chains are also important if you use third party backup management products. Depending on the used vendor, log chains are incorporated in the solution to ensure uniqueness of archived logs. The log chain information for every log file is also written to the database recovery history file. For every log file, we see the archive location as well as the log chain. In the case of a database recovery, the information from the history file is used by the database to locate the log files that are required for the recovery. To access the information regarding the archived logs in the database history file, you can use the LIST HISTORY ARCHIVE LOG command that is described in section 4.3.

3 Backup and Recovery – Details

As described at the beginning of the document, backup and recovery is a fully integrated part of the DB2 for Linux, UNIX and Windows database engine. The DB2 BACKUP command saves all the information that is necessary to rebuild the database in the case of a failure. To recover data, you can use the RESTORE and ROLLFORWARD utilities. The following chapter introduces the different commands that are available for data recovery. In addition to the BACKUP, RESTORE, RECOVER and ROLLFORWARD utilities, we also introduce the tools that are available to monitor these operations.

We will not provide a full syntax of each command but rather concentrate on an overview and examples on how you can use these commands. For a full description of the commands, see the IBM DB2 Information Center on the Internet.

3.1 BACKUP DATABASE

To backup the complete database including all table spaces, you can use the BACKUP DATABASE command. Alternatively, the database administrator can decide to back up single table spaces of the database.

You can use the BACKUP DATABASE command of DB2 to backup a database offline or online. An offline backup establishes an exclusive connection to the database and no other applications can access the database concurrently. During an online backup, applications can work on the database. Some other utilities, for example, REORG TABLE, are incompatible to an online backup. For more information about the complete overview of compatible utilities, see the IBM DB2 Information Center.

By default, the DB2 backup utility saves all pages of the table spaces to the backup media. In addition, the DB2 database management system supports incremental backups. Here the utility saves only these pages that have changed since the most recent successful backup. Depending on the keywords used, the most recent backup can be any database backup (INCREMENTAL DELTA) or a full backup (INCREMENTAL).

The DB2 backup utility can include the range of log files required to restore and roll forward the backup image to some consistent point in time. As of DB2 9.5 this is the default for online backups. A variety of different target media is supported for DB2 backup (and recovery) operations, for example, disk, tape, storage snapshots, IBM Tivoli Storage Manager or third party tools (for example, EMC NetWorker or HP OpenView Data Protector).

The DB2 BACKUP command syntax has three main areas:

1) The scope of the backup:
   - Name of the database
   - Online / offline
   - Full database backup or backup of one or more table spaces
   - Database partition
   - Full or incremental backup
2) The target for the backup image (Disk, Tape, Pipe, Tivoli Storage Manager, XBSA, Vendor Library, Snapshot)
3) Additional backup options
   - Number of buffers, buffer size, parallelism of backup (when not specified, the database engine will automatically tune the backup)
   - Compression of the backup image (for example, to save disk space) using the DB2 built-in or a third party compression library
   - Impact of the BACKUP command (whether to use throttling)
   - Inclusion or exclusion of database logs necessary for recovery in the backup image

The following list explains how these parts of the backup command are combined to start a database backup:

1) First, we define the scope: online backup of a database called P01:
   BACKUP DB P01 ONLINE <…>
2) Now, in the second part, we define Tivoli Storage Manager as the target of the backup – with two sessions on the Tivoli Storage Manager server:
   <…> USE TSM OPEN 2 SESSIONS <…>
3) Finally, in the third part, we throttle the backup utility and exclude log files from the image:
   <…> UTIL_IMPACT_PRIORITY 50 EXCLUDE LOGS

A complete description of the different options of the DB2 BACKUP command would go beyond the scope of this document. For more information about the complete DB2 BACKUP command syntax, see the IBM DB2 Information Center.

To provide a better understanding of the DB2 BACKUP command, the following section contains some examples:

- Offline backup of a database named SAMPLE to a Windows directory called d:\backups

  ```
  BACKUP DATABASE SAMPLE TO d:\backups
  ```

- Online backup of database WGN to IBM Tivoli Storage Manager:

  ```
  BACKUP DB WGN ONLINE USE TSM
  ```

- Online backup of database ISP to EMC NetWorker using twelve tape drives; the parameterization of the NetWorker Module is done through an options file:

  ```
  BACKUP DB ISP ONLINE LOAD /usr/lib/libnsrdb2.so OPEN 12 SESSIONS
  OPTIONS @/db2/db2isp/.nsrenv
  ```

- Online backup of two table spaces of database Q43 to two file systems /data/backups1 and /data/backups2:

  ```
  BACKUP DB Q43 TABLESPACE (Q43#BTABD, Q43#BTABI) ONLINE TO
  /data/backups1, /data/backups2
  ```

- Online backup of partition 7 of database GBP to IBM Tivoli Storage Manager

  ```
  BACKUP DB GBP ON DBPARTITIONNUM 7 ONLINE USE TSM
  ```

- Incremental delta online backup (backup of all changed database pages since the most recent backup) of database Q34 to tape:

  ```
  BACKUP DB Q34 ONLINE INCREMENTAL DELTA TO /dev/rmt0
  ```

Note that the BACKUP command supports both the keyword DATABASE and the short form DB. This abbreviation can also be used for commands RESTORE, ROLLFORWARD and RECOVER.

You can interrupt a running DB2 backup operation any time by using the DB2 FORCE APPLICATION command. This command forces the agent that is managing the backup process. For more information about how to interrupt a running backup, see section 4.7 “

LIST APPLICATIONS / FORCE APPLICATION".

3.2 RESTORE DATABASE

The RESTORE command of DB2 is used to restore a DB2 backup image that was created using the DB2 BACKUP command. The RESTORE command can recreate either a full database or a subset of table spaces. To restore single table spaces, a full database backup image can be used. The REBUILD option of the RESTORE command allows you to restore a full database out of a set of table space backups.

If you restore a database using incremental backup images, the RESTORE command automatically restores all necessary full and incremental backup images to recreate the database. You can use the RESTORE command can to restore a history file of a database from a backup image as well as to generate a script file for a redirected restore operation.

In addition, you can also perform cross-platform restores with DB2. All big Endian UNIX platforms are compatible as well as the little Endian platforms. You cannot, however, restore incremental backup images if the operating system of the source system is different from the operating system of the target system.

To upgrade a database from one DB2 version to a higher one, you can use an offline backup image. For example, from a DB2 instance running on DB2 Version 9.7 you can restore a backup image that was taken with DB2 Version 9.1. At the end of the restore operation, the database upgrade is invoked automatically.

To provide a better understanding of the command syntax, we can summarize the DB2 RESTORE command to its main sections – similar to the BACKUP command:

1) The scope of the restore:
   - Name of the database to be restored
   - Restore of the full database or of a set of table spaces or of the history file only?
   - Restore of the log files included in the backup image?
   - Should the rebuild method be used to restore a full database out of table space backups?
   - Are we restoring from incremental backups?
2) The location of the backup image to be restored:
   - Disk, Tape, Pipe, Tivoli Storage Manager, XBSA, Vendor Library, Snapshot
   - Timestamp of the backup image
3) The target of the restore
   - Restore into an existing database with the same name?
   - Or restore into a new database with a different alias?
   - Do we want to change the database directory or storage paths (for automatic storage)?
   - When restoring log files provide the target directory where the files will be restored to
   - Definition of a new log path for the database
4) Additional restore options:
   - Number of buffers, buffer size, parallelism of backup (when not specified, DB2 will automatically tune the restore)
   - Whether to replace an existing history file?
   - Is this a redirected restore (new definition of table space containers)?
   - Name of the decompression library in case the backup image was compressed with a third party compression library

The following explains how the four parts of the restore command are combined together to start a database restore:

1) First, we define the scope: a restore of the database P01:
   RESTORE DB P01 <…>
2) Now, in the second part, we define the backup image to be used for the restore; in our example the backup was performed on 2009/06/15 and was stored in IBM Tivoli Storage Manager:
   <…> USE TSM TAKEN AT 20090615 OPEN 2 SESSIONS <…>
3) In the third part, we define two new automatic storage paths for the database as well as a new log path:
   <…> ON /db2/P01/sapdata1, /db2/P01/sapdata2 NEWLOGPATH /db2/P01/log_dir <…>

4) Finally, some additional restore options – here just to overwrite the existing database without prompting for a confirmation:
   <…> WITHOUT PROMPTING

A complete description of the different options of the DB2 RESTORE command would go beyond the scope of this document. For information about the complete DB2 RESTORE command syntax, see the IBM DB2 Information Center.

To provide a better understanding of the DB2 RESTORE command, the following list contains a few examples:

- Restore the most recent backup image of database SAMPLE that is available in the Windows directory d:\backups:

  ```
  RESTORE DB SAMPLE FROM d:\backups
  ```

- Restore of database WGN using a backup image taken at 2009/02/22 from IBM Tivoli Storage Manager :

  ```
  RESTORE DB WGN USE TSM TAKEN AT 20090222
  ```

- Restore of database ISP from EMC NetWorker using twelve tape drives and parameterization of the NetWorker Module through an options file

  ```
  RESTORE DB ISP LOAD /usr/lib/libnsrdb2.so OPEN 12 SESSIONS OPTIONS
  @/db2/db2isp/.nsrenv TAKEN AT 20081227150408
  ```

- Rebuild of database ECA with three table spaces from a backup image on Tivoli Storage Manager:

  ```
  RESTORE DB ECA REBUILD WITH TABLESPACE (SYSCATSPACE, ECA#BTABD,
  ECA#BTABI) USE TSM
  ```

If you want to use the DB2 RESTORE command, you have to consider the following:

- The format for the timestamps used by the TAKEN AT parameter is <yyyymmddhhmmss>. A full or a partial timestamp can be provided. The full timestamp of the backup image is, for example, returned on the successful completion of the backup.
- If you are using partial timestamps in the TAKEN AT parameter make sure the DB2 database only finds one backup image for the given date/time. Otherwise, the RESTORE command will fail. For example, assuming database WGN was backed up twice on 2009/02/22 – at 8:15h and 18:30h. In this case a timestamp in the format TAKEN AT 20090222 would not be sufficient (since DB2 database sees two backup images for the day). Instead, a timestamp has to be provided that at least includes the hour so that the DB2 utility only sees one backup image. For example, use something like TAKEN AT 2009022208 for the backup from the morning.
- If no backup image timestamp is specified, the DB2 database will automatically use the most recent successful backup image that is available at the location.
- In standard SAP environments, every table is defined with a separate index table space. If you want to restore only a sub set of table spaces, a complete set of table spaces has to be restored. If, for example, table space <SAPSID>#POOLD is restored, the index table space <SAPSID>#POOLI has to be restored as well. Otherwise the restore operation fails.

3.3 Redirected Restore

As stated earlier, the DB2 BACKUP command saves the database configuration including the table space layout. The RESTORE command will use this information to create the database with the exact same name, configuration and table space layout of the source database. For some cases, this default behavior of the DB2 restore is not desired. For example, if a copy of an existing database is created, usually the database name is changed, too. To change the database name, specify the INTO option of the RESTORE command as shown in the following example.

- Restore the image of database FCB into a new database named FCK:
  `RESTORE DB FCB USE TSM TAKEN AT 20090222 INTO FCK`

The command above restores database FCB into to a new database named FCK. However, the new FCK database would, for example, still have the same table space container paths as the source database FCB. In many cases, this is not desired – especially in SAP environments, where the database name <DBSID> is part of the DB2 related file systems (like /db2/<DBSID>/sapdata1). This section describes how you can use the DB2 RESTORE command to change the database layout.

3.3.1 REDIRECT RESTORE Overview

The DB2 RESTORE supports so-called redirected restore operations. A redirected restore is performed if table space containers are restored in a different physical location. Also, the restore of a database to a target server different than the source database server is considered to be a redirected restore. A redirected restore is determined by using the keyword REDIRECT as part of the RESTORE command.

A redirected restore has the following phases:

1) Issue the RESTORE DATABASE command with the REDIRECT option
   The DB2 database management system reads the configuration of the source database from the backup image.
2) To define the table space containers of the new database, use the SET TABLESPACE CONTAINERS command
   The DB2 database management system creates and formats an “empty” database using the newly defined table space containers.
3) Issue the RESTORE command again, this time using the CONTINUE option
   The DB2 database management system now writes data into the new table space containers.

Once the restore has finished successfully, the database is put into rollforward recovery pending state. To apply the log files from the source database and to bring the new database online, you can use the ROLLFORWARD command.

3.3.2 REDIRECT RESTORE Scripts

When performing a redirected restore it is best practice to put all necessary commands into a script. Creating such a script manually can be a tedious work, especially on large SAP databases with many table space containers that need to be defined. To simplify the process, the DB2 database management system offers the option to automatically generate a script for redirect restores using the RESTORE command. In addition to the DB2 functionality, SAP ships a tool called brdb6brt with similar functionality. Both utilities generate the scripts that are required to perform a redirected restore.

To remain in the scope of this document, we will only roughly describe both tools but we cannot provide details. A comprehensive description of redirected restores will be included in the best practices and advanced topics paper of this series.

Both tools – the DB2 RESTORE and the brdb6brt – query the table space information of the source database. The way how this data is gathered is different: The RESTORE command retrieves the information directly from the specified backup image. Therefore, the table space information in the generated script matches the layout that is stored in the backup image. The brdb6brt tool queries the current table space layout of the active source database. If table spaces have been created or dropped since the last backup, you cannot use the brdb6brt tool to create a redirected restore script.

Since the tool reads the current table space layout the script would contain information about table spaces that are not contained in the backup image.

Using the DB2 RESTORE command to create the redirected restore script is useful if the source database is not accessible, for example, in case of a disaster recovery. The brdb6brt tool on the other hand provides additional functions. For example, in addition to creating a script file for redirected restores, you can use the tool to start DB2 backups and restores as well as to create a configuration file for the db2relocatedb command. In addition, the brdb6brt tool is able to automatically rename all occurrences of the source database name to match the new database name.

The DB2 RESTORE command accesses the specified backup image and reads the table space configuration. This information is used for the SET TABLESPACE CONTAINERS part of the redirected restore script. The following example shows how you can create a redirected restore script using the DB2 RESTORE command. In our example, a script for restoring database XEG into YEG is created. The backup was created using the EMC NetWorker library (libnsrdb2.so):

```
RESTORE DB XEG LOAD /usr/lib/libnsrdb2.so OPEN 2 SESSIONS INTO YEG REDIRECT
GENERATE SCRIPT xeg2yeg.clp
```

The RESTORE command generates a script xeg2yeg.clp that is based on the information of the database XEG. You can edit this script to reflect the changes necessary for a restore to the target database YEG. Once all the necessary changes have been made in the script, you can transfer it to the target DB2 instance. On the target DB2 instance, you can execute using the script using the DB2 command line processor (CLP). The script automatically creates a log file for the process (in our example XEG_NODE0000.out). To execute the redirected restore script, use the following command:

```
usdbyeg:db2yeg % db2 –tvf xeg2yeg.clp
```

The program brdb6brt is shipped as part of the SAP NetWeaver ABAP kernel. It is located in the /sapmnt/<SAPSID>/exe directory. You can use the command similar to the REDIRECT GENERATE SCRIPT option of the DB2 RESTORE command. In our example, you create a new database YEG out of the XEG database by executing the brdb6brt tool with the following options:

```
uscixeg:xegadm % brdb6brt –bm RETRIEVE –es –replace XEG=YEG,db2xeg=db2yeg
```

By executing the brdb6brt, a script file called '<DBSID>_NODE<nnnn>.scr' is created; in our example it is called XEG_NODE0000.scr. Unlike the DB2 RESTORE functionality, the brdb6brt tool does not read a backup image on the backup storage but from the source database. The –replace option automatically replaces the old SAP system ID (XEG) with the new one (YEG). Once the script file has been adapted to the system environment, you have to transfer it to the target database server. You can start the script using the DB2 command line processor. Just like with the script that was created using the RESTORE command, an output file is automatically written, containing all the output of the different DB2 commands that were called by brdb6brt.

3.4 ROLLFORWARD DATABASE

If a recoverable DB2 database is restored, the database is set to rollforward pending state at the end of the restore. No connect to the database is possible at this point. To open the database, you have to use the ROLLFORWARD command.

To apply transaction logs to a database in rollforward pending state, the DB2 database management system provides the ROLLFORWARD command. You can use this command to roll forward a database to a specified point in time, to the end of the available log files, or to the minimum recovery time (end of backup). The point in time can be specified as Coordinated Universal Time (UTC) or the local time zone, which is set on the database server. The format used for the time stamp in the ROLLFORWARD command is different than the one used for RESTORE. For ROLLFORWARD, it has the format <yyyy-mm-dd-hh.mm.ss>.

You can issue the ROLLFORWARD command can be issued multiple times to reach different (more recent) points in times, for example, on a standby database. Once a point in time has been provided, you cannot start another iteration of the ROLLFORWARD command with an older time stamp.

As soon as the STOP (or the equivalent COMPLETE) parameter of the ROLLFORWARD command has been issued the DB2 engine rolls back all transactions that were not committed and opens the

database. All subsequent ROLLFORWARD commands that were submitted against this database will then fail.

A complete description of the different options of the DB2 ROLLFORWARD command would go beyond the scope of this document. For more information, see the IBM DB2 Information Center.

The following list contains a few examples of the DB2 ROLLFORWARD command:

- Query the Rollforward state of database ISP (for example, to see the next log file required for recovery):

  ```
  ROLLFORWARD DB ISP QUERY STATUS
  ```

- Rollforward of a database named SAMPLE to end of logs and open the database at the end of the operation:

  ```
  ROLLFORWARD DB SAMPLE TO END OF LOGS AND STOP
  ```

- Rollforward of database WGN to 02/23/2009 12:00 (UTC):

  ```
  ROLLFORWARD DB WGN TO 2009-02-23-12.00.00.000000
  ```

3.5 RECOVER DATABASE

The RECOVER DATABASE command simplifies database recover operations by combining the RESTORE and the ROLLFORWARD commands. The database administrator has only to specify the required point in time of the recovery. All actions that are necessary to reach this state, for example, finding the proper backup image(s), restore and roll forward recovery, are then automatically performed by the RECOVER command.

The RECOVER command reads the history file of the database. In case, this file is not available – for example, where the target database does not exist – the RECOVER command allows using a defined history file. You can restart an aborted RECOVER command. If possible, the DB2 engine will attempt to continue the previous RECOVER.

The following are a few examples of the DB2 RECOVER command:

- Recovery of a database named SAMPLE to the end of logs:

  ```
  RECOVER DB SAMPLE
  ```

- Recovery of a database named WNG to a specified point in time (UTC):

  ```
  RECOVER DB WNG TO 2009-03-30-12.00.00
  ```

For more information about the complete RECOVER command syntax, see the IBM DB2 Information Center.

4 Additional Commands and Utilities

The following section provides information about additional useful commands and utilities in the context of logging, backup and recovery. These commands are used mainly to monitor the DB2 utilities and provide useful information for the day-to-day operation. We also describe how you can identify a running DB2 backup and how you can use this information to cancel a backup operation by forcing off the backup session from the database.

4.1 DB2CFEXP / DB2CFIMP

A full DB2 database backup includes everything that is required to rebuild the database, for example, data, log files and configuration information. However, information outside the database, for example, the DB2 registry variables or the configuration of the database manager (instance) is not part of the database backup.

The DB2 database management system offers commands to export (CFEXP) and import (DB2CFIMP) client configurations. In addition, you can use these commands to take a backup of the database manager configuration, the DB2 registry settings and the database and node directories. The DB2CFEXP command writes the information into a text file that you can store in a secure location. In case of a recovery, you can import the file can be using the DB2CFIMP command.

The following are example commands that you can use to back up the database configuration:

- Create a backup of the information located outside the database to a file name cfg_backup.txt:
  `db2cfexp cfg_backup.txt BACKUP`
- Import the information from a file created by db2cfexp:
  `db2cfimp cfg_backup.txt`

4.2 ARCHIVE LOG

The DB2 log manager automatically archives and retrieves DB2 log files. Under certain scenarios, for example, for standby databases, it might be advantageous to force an archive log operation at a certain point in time. DB2 for Linux, UNIX and Windows supports this for recoverable databases using the ARCHIVE LOG command.

By issuing this command, the DB2 engine closes and truncates the currently active log file. This log file is then handled by the method described in database configuration parameter LOGARCHMETH1.

The following example would close the currently active log for database Q43 and move the log header to the next log file. The log file is archived according to the method defined in database configuration parameter LOGARCHMETH1:

```
ARCHIVE LOG FOR DB Q43
```

Before you issue the ARCHIVE LOG command you have to disconnect your current DB2 CLP session from the database (for example, by using the TERMINATE command). Otherwise, the ARCHIVE LOG will fail with SQL message SQL1493N.

In addition to the manual ARCHIVE LOG command, the database management system truncates (and archives) under certain conditions the current active log. For example, at the end of an online backup the current active log is archived. This allows collecting the set of log files required to recover the database. If the DB2 9.5 default is used, these log file(s) are included in the backup image.

4.3 LIST HISTORY

The DB2 LIST HISTORY command scans the database history file. This file contains information about the execution of the different utilities that are available, for example, BACKUP and ROLLFORWARD. The BACKUP keyword of the LIST HISTORY command lists backup and restore operations. You can use the information provided in the LIST HISTORY command, for example, to check whether a backup operation was executed successfully and how long the operation was running. The output will provide additional information about the backup, for example, the location of the backup image and the type of the backup operation (offline, online, incremental)

You can also access the information that is available in the history file via SQL using the SYSIBMADM.DB_HISTORY view.

It is possible to query the complete history file of a database or to only display the entries after a certain point in time. The following examples show both possible methods:

- To list all backup and restore operations for database Q43, enter:
  `LIST HISTORY BACKUP ALL FOR DB Q43`
- To list backup and restore operations since 03/15/2009 for database ECA, enter:
  `LIST HISTORY BACKUP SINCE 20090315 FOR DB ECA`

In following example output we see an entry for a successful online backup:

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20090327182729001   N    O  S0037420.LOG S0037476.LOG
 ------------------------------------------------------------------------------
  Contains 32 tablespace(s):

  00001 SYSCATSPACE
  00002 Q43#BTABD
  00003 Q43#BTABI
  00004 SYSTOOLSPACE
[...]
  00031 Q43#ODSD
  00032 Q43#ODSI
 ------------------------------------------------------------------------------
    Comment: DB2 BACKUP Q43 ONLINE
 Start Time: 20090327182729
   End Time: 20090328102304
     Status: A
 ------------------------------------------------------------------------------
  EID: 60443 Location: /usr/lib/libnsrdb2.so
```

The next example shows the output for an aborted backup operation. The backup of database OPT failed with SQL message -2413 (as seen in the SQLCA area). The SQL message was thrown because the database is using circular logging and therefore online backups are not possible.

```
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20090327161157000   N       S0000003.LOG
 ------------------------------------------------------------------------

 ------------------------------------------------------------------------
    Comment: DB2 BACKUP OPT ONLINE
 Start Time: 20090327161157
   End Time: 20090327161202
     Status: A
 ------------------------------------------------------------------------
  EID: 721 Location:

SQLCA Information

 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: -2413   sqlerrml: 0

 sqlerrmc:
 sqlerrp : sqlubIni
 sqlerrd : (1) 0                (2) 0                (3) 0
           (4) 0                (5) 0                (6) 0
 sqlwarn : (1)       (2)       (3)       (4)        (5)        (6)
           (7)       (8)       (9)       (10)       (11)
 sqlstate:
```

Similar to the information provided about backup and recovery operations, you can use the DB2 history file to query information about the archival of log files.

```
LIST HISTORY ARCHIVE LOG ALL FOR DB <db>
```

The following example output shows the entries for a log file that was archived to disk. Log file S0000009.LOG was archived to the disk location /db2/EK1/archived_logs. The log file belongs to log chain number 3 – as shown in the column “Current Log”: C0000003.

```
 Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  X  D  20090304152433      1    D  S0000009.LOG C0000003
 ------------------------------------------------------------------------

 ------------------------------------------------------------------------
    Comment:
 Start Time: 20090304152433
   End Time: 20090304152433
     Status: A
 ------------------------------------------------------------------------
  EID: 31 Location: /db2/EK1/archived_logs/db2ek1/EK1/NODE0000/C0000003/S0000009.LOG
```

4.4 LIST UTILITIES

The LIST UTILITIES command allows you to monitor DB2 utilities like BACKUP, RESTORE or ROLLFORWARD. This command lists all currently active DB2 utilities on all database partitions. To display additional information about the utility, for example, how many bytes have already been read, you can use the SHOW DETAIL parameter.

You can also access the information provided by the LIST UTILITIES command via SQL. The DB2 dabase management system provides the following two database views for this purpose: SYSIBMADM.SNAPUTIL_PROGRESS and SYSIBMADM.SNAPUTIL.

The following output shows the information that is returned by the LIST UTILITIES SHOW DETAIL command. We see that an unthrottled online backup of database ICV is active. Approximately 80 percent of the database is already backed up:

```
icvdb:db2icv 1> db2 list utilities show detail

ID                                = 23487
Type                              = BACKUP
Database Name                     = ICV
Partition Number                  = 0
Description                       = online db
Start Time                        = 03/30/2009 11:02:49.114974
State                             = Executing
Invocation Type                   = User
Throttling:
   Priority                       = Unthrottled
Progress Monitoring:
   Estimated Percentage Complete = 80
      Total Work                  = 2332814791685 bytes
      Completed Work              = 1855671877101 bytes
      Start Time                  = 03/30/2009 11:02:49.115710
```

4.5 GET SNAPSHOT

The DB2 GET SNAPSHOT command collects different types of status information using the monitoring facilities of the DB2 database management system. For the scope of our document, you can use the database snapshot to access some interesting information, for example:

- Last backup timestamp
- File number of first, current and last active log
- Log space used by the database and number of secondary log files allocated.

The following example shows excerpts from the result of a database snapshot that was taken on database Q43. Looking at the output, we can easily identify the timestamp of the last successful database backup of the database (06/19/2009 06:28:01). The next section of the output helps you to understand the amount of log space that is used by your database. For example, if the snapshot constantly reports that secondary log files are allocated, the number primary log files should be increased. Finally, in the last section of the output, information about the current log files of the database is returned:

```
usdbq43:db2q43 % db2 get snapshot for db on q43

             Database Snapshot

Database name                               = Q43
Database path                               = /db2/Q43/db2q43/NODE0000/SQL00001/
<…>
First database connect timestamp            = 06/16/2009 15:53:35.057275
Last reset timestamp                        =
Last backup timestamp                       = 06/19/2009 06:28:01.000000
Snapshot timestamp                          = 06/22/2009 11:02:05.600354
<…>
Log space available to the database (Bytes)= 25966749471
Log space used by the database (Bytes)     = 145250529
Maximum secondary log space used (Bytes)   = 0
Maximum total log space used (Bytes)       = 4216103622
Secondary logs allocated currently         = 0
<…>
File number of first active log            = 41373
File number of last active log             = 41417
File number of current active log          = 41374
File number of log being archived          = Not applicable
<…>
```

4.6 db2pd

The db2pd command offers database administrators a variety of very useful monitoring information. The information provided by the db2pd command is similar to the data that you can gather by using the GET SNAPSHOT or LIST UTILITIES commands.

The following list provides a few useful db2pd parameters in the scope of backup and recovery:

- `-recovery -db <db name>`
  Shows the progress made by a recovery process
- `-logs -db <db name>`
  Lists the active log files of a database and the status of the log file archiving
- `-fvp <agent eduid of backup process>`
  Controls a third party vendor process that is for example, performing a backup
- `-utilities`
  Information similar to output provided by the LIST UTILITIES command

Note that unlike the LIST UTILITIES command, db2pd only lists information regarding the current database partition. In case you want to retrieve information for all or one specific database partition use the appropriate options (`-dbp <partition>` or `-alldbp`).

The following examples show the different outputs of db2pd if you use different options related to backup and recovery:

1) Output of db2pd –utilities: We see information returned regarding an active roll forward recovery on database ICY; the rollforward operation is in the first phase (forward recovery):

```
icydb:db2icy > db2pd -utilities

Database Partition 0 -- Active -- Up 247 days 12:24:24

Utilities:
Address            ID         Type                    State      Invoker    Priority
StartTime           DBName   NumPhases  CurPhase   Description
0xC00000001137FBA0 2233       ROLLFORWARD RECOVERY   0          0          0          Tue Mar
31 11:31:24 ICP      2           1           Database Rollforward Recovery

Progress:
Address            ID         PhaseNum   CompletedWork               TotalWork
StartTime           Description
0xC00000001137E818 2233       1          16415179148 bytes           Unknown
Tue Mar 31 11:31:24 Forward
0xC00000001137E930 2233       2          0 bytes                     Unknown
NotStarted          Backward
```

2) Output of db2pd –recovery: This output shows related information to the rollforward process on database ICY that we saw in the previous example. We get the Log Sequence Number (LSN) that the rollforward process is currently processing. The database log file holding the LSN (S0134711.LOG) is displayed as well:

```
hs2050:db2icy > db2pd -recovery -db icy

Database Partition 0 -- Database ICY -- Active -- Up 0 days 00:18:16

Recovery:
Recovery Status     0x04000401
Current Log         S0134711.LOG
Current LSN         262BA033DDB6
Job Type            ROLLFORWARD RECOVERY
Job ID              2233
Job Start Time      (1238491884) Tue Mar 31 11:31:24 2009
Job Description     Database Rollforward Recovery
Invoker Type        User
Total Phases        2
Current Phase       1

Progress:
Address            PhaseNum   Description          StartTime           CompletedWork
TotalWork
0xC00000001137E818 1          Forward              Tue Mar 31 11:31:24 16592958914 bytes
Unknown
0xC00000001137E930 2          Backward             NotStarted          0 bytes
Unknown
```

3) Output of db2 –db Q43 –log: We see the logging status of database Q43, the current active log file (41374) as well as the log archive status (here Success). At the end we see all log files allocated in the log directory at this time including the LSN of the log files.

```
usdbq43:db2q43 % db2pd -db q43 -log | more


Database Partition 0 -- Database Q43 -- Active -- Up 5 days 19:24:13


Logs:
Current Log Number            41374
Pages Written                 41252
Method 1 Archive Status       Success
Method 1 Next Log to Archive  41374
Method 1 First Failure        n/a
Method 2 Archive Status       n/a
Method 2 Next Log to Archive  n/a
Method 2 First Failure        n/a

Address            StartLSN       State      Size       Pages      Filename
0x000001000D17CE18 0x07E85C1C8000 0x00000000 51200      51200      S0041373.LOG
0x0000010845BC6858 0x07E8689C8000 0x00000000 51200      51200      S0041374.LOG
0x00000100006C88F8 0x07E8751C8000 0x00000000 51200      51200      S0041375.LOG
0x00000100006C89B8 0x07E8819C8000 0x00000000 51200      51200      S0041376.LOG
0x000001000D14F458 0x07E88E1C8000 0x00000000 51200      51200      S0041377.LOG
0x00000100006C8BF8 0x07E89A9C8000 0x00000000 51200      51200      S0041378.LOG
0x000001000D157758 0x07E8A71C8000 0x00000000 51200      51200      S0041379.LOG
<…>
```

4.7 LIST APPLICATIONS / FORCE APPLICATION

You can interrupt a running DB2 BACKUP or RESTORE operation any time by using the DB2 FORCE APPLICATION command. The FORCE command requires the application handle of the DB2 process that performs the DB2 backup. To identify this handle, use, for example, the DB2 LIST APPLICATIONS command (with the SHOW DETAIL command option).

The following example shows how you can identify the DB2 application that is performing the backup:

1) In the first step, we use the LIST APPLICATIONS SHOW DETAIL command to find the backup operation (look for the “Performing a Backup” keyword)

```
usdbxeg:db2xeg % db2 list applications show detail | grep -i backup
DB2XEG
db2bp                18471      *LOCAL.db2xeg.090512103634
00001 33         0                12751           Performing a Backup
05/12/2009 12:36:35.467379 XEG      /db2/XEG/db2xeg/NODE0000/SQL00001/
```

2) From the first step, we now know the application handle of the db2bp process that is performing the backup. In the next step, we now force this application handle 18471 by using the DB2 command FORCE APPLICATION:

```
usdbxeg:db2xeg % db2 "force application (18471)"
DB20000I  The FORCE APPLICATION command completed successfully.
DB21024I  This command is asynchronous and may not be effective immediately.
```

3) Forcing off an application from the database is performed asynchronously and might take some time. The application itself receives an SQL1224N message. Assuming the backup operation was started in a DB2 command line window, the message looks as shown in the following example:

```
usdbxeg:db2xeg % db2 "backup db XEG online   load /usr/lib/libnsrdb2.so open 2 sessions
options @/db2/XEG/admin/backup.opt exclude logs"
SQL1224N  The database manager is not able to accept new requests, has
terminated all requests in progress, or has terminated your particular request
due to an error or a force interrupt.  SQLSTATE=55032
```

5 DB2 and SAP Integration

The following chapter describes the integration of the DB2 backup functionality into the SAP DBA Cockpit. The DBA Cockpit is the central database monitoring and administration component of SAP systems. In the DBA Cockpit – besides many other DB2-related functions – you can schedule DB2 database backups and monitor their status. The DBA Cockpit also allows checking the archiving of DB2 transaction log files.

5.1 Configuration of Backup and Logging

The DBA Cockpit provides a graphical interface to the DB2 BACKUP DATABASE command and the configuration that was described in the previous sections. You can use the DBA Cockpit to configure DB2 logging and backups.

To start the DBA Cockpit, call transaction DBACOCKPIT in your SAP system. In the navigation frame, choose *Configuration* → *Database* and choose the appropriate section that you want to configure. In case of the parameters for logging, backup and recovery, use the options *Logging*, *Log File Management*, *Backup & Recovery* and *Tivoli Storage Manager*.

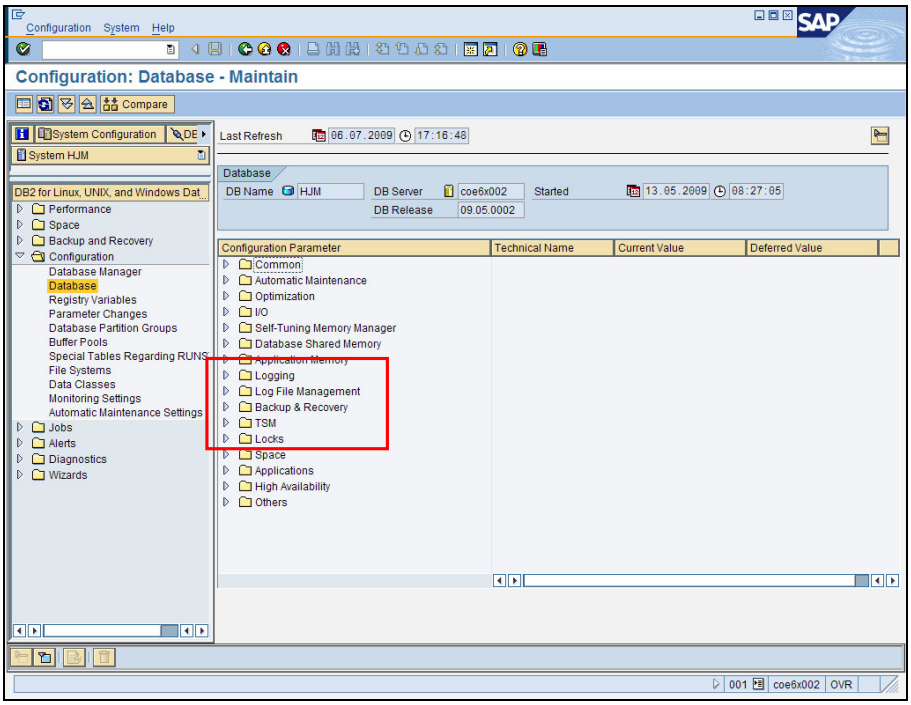


Figure 5-1: DBACOCKPIT – Database Configuration

To change a parameter, double-click the parameter and change the value as required. Some of the parameters are informational and cannot be changed. The following Figure 5-2 shows, for example, the configuration parameters that are related to database logging:

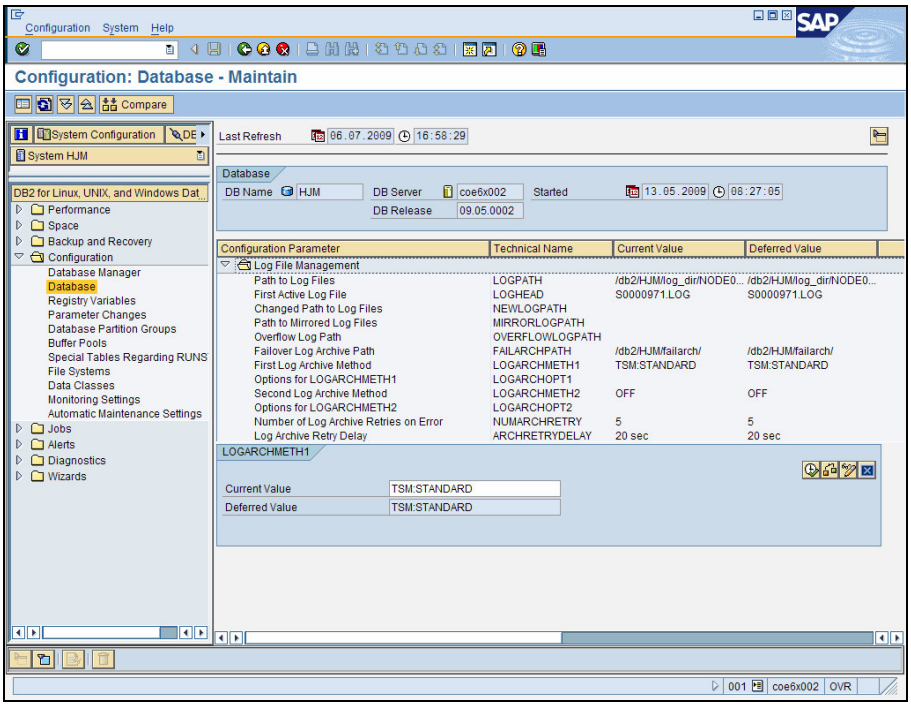


Figure 5-2: Parameter Changes with the DBACOCKPIT

5.2 Performing Backups

You can start and schedule ad hoc database backups as well as recurring backups using the DBA Cockpit in your SAP system. Newer versions of the DBA Cockpit provide an option to configure the DB2 automatic backups. To manually perform or schedule a database backup choose Jobs → DBA Planning Calendar in the navigation frame of the DBA Cockpit. The DBA Planning Calendar screen appears as shown in the following figure:

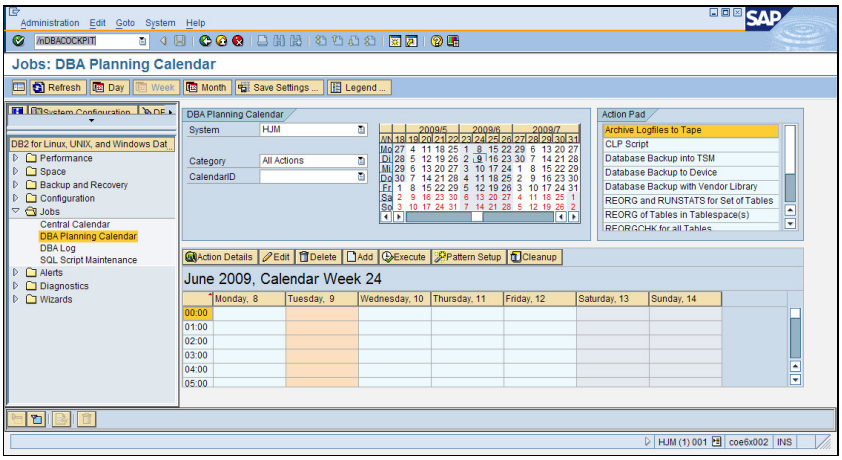


Figure 5-3: DBA Planning Calendar

The Action Pad on the DBA Planning Calendar screen offers a list of DB2 administration and maintenance options. The following options are available to perform backups:

- *Database Backup into TSM*
  This action performs a database backup into IBM Tivoli Storage Manager
- *Database Backup to Device*
  Performs a database backup to disk, onto tape or into a named pipe

- *Database Backup with Vendor Library*
  Uses a Vendor library like EMC NetWorker to perform a backup

To schedule backups, drag and drop the corresponding action item into the calendar area. The target calendar cell represents the combination of desired execution date and execution time. The following figure shows how you can schedule a DB2 backup into IBM Tivoli Storage Manager at 14:00h on Tuesday, June, 9th.

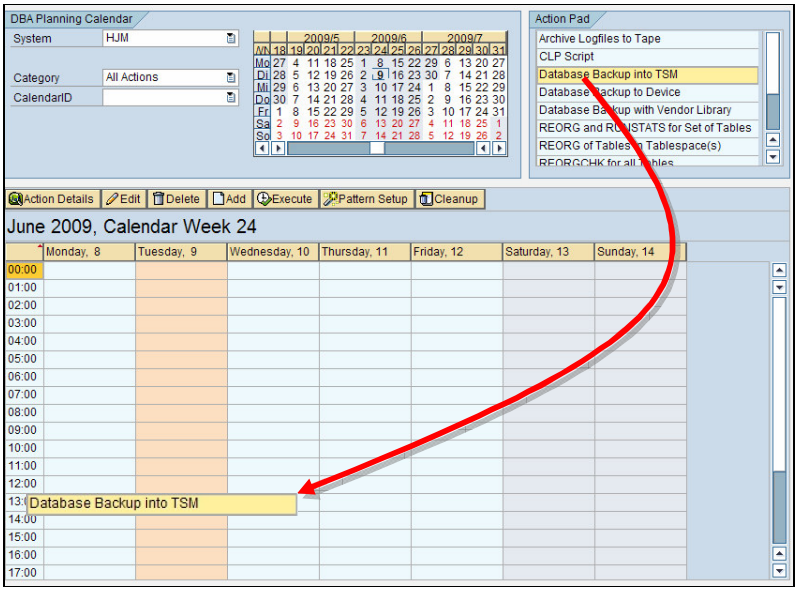


Figure 5-4: Scheduling a Backup using Drag and Drop

Alternatively, choose the desired calendar cell and choose the Add pushbutton or simply double click the required calendar cell. Either way, a dialog box appears.

If you have used the drag and drop approach to schedule an action, the respective job is already preselected in the Action field. If you used one of the alternative options to schedule backups, you have to manually select the required job in the Action field.

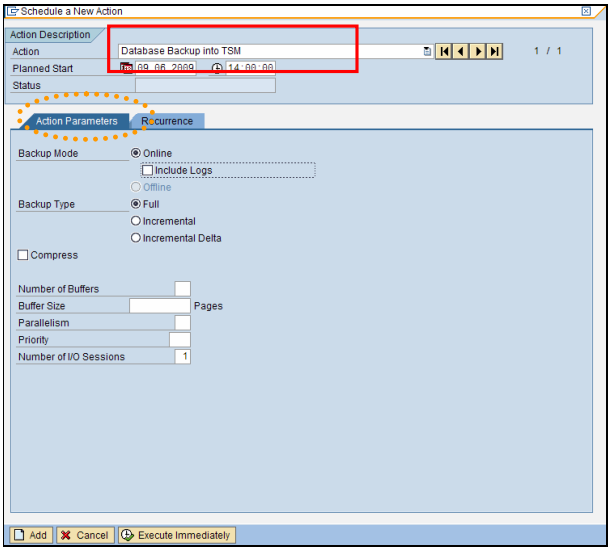


Figure 5-5: Action Dialog Box – Action Parameters

On the Action Parameters tab page of the Schedule a New Action dialog box, you can set the required parameters for the DB2 BACKUP command. The available options are more or less the same options that are available on the DB2 command line. The only limitation is that all backups scheduled via the DBA Planning Calendar are online backups.

You can set the following backup parameters:

- Including Log Files into the backup image (the INCLUDE LOGS parameter of the BACKUP command) – not required with DB2 9.5 and higher
- Full or Incremental Backups (INCREMENTAL or INCREMENTAL DELTA)
- Backup Compression (the COMPRESS parameter of the BACKUP command)
- Number of buffers, buffer size, parallelism – Usually, you can leave these fields empty to let the DB2 database management system automatically tune the parameters
- Priority defines whether DB2 should throttle the backup utility – leave this field empty to run the backup unthrottled
- Number of I/O sessions (the OPEN <n> SESSIONS parameter of the backup) – use a value that fits the backup infrastructure

On the Recurrence tabpage of the Schedule a New Action dialog box, you can define when and how often the backup is supposed to run.

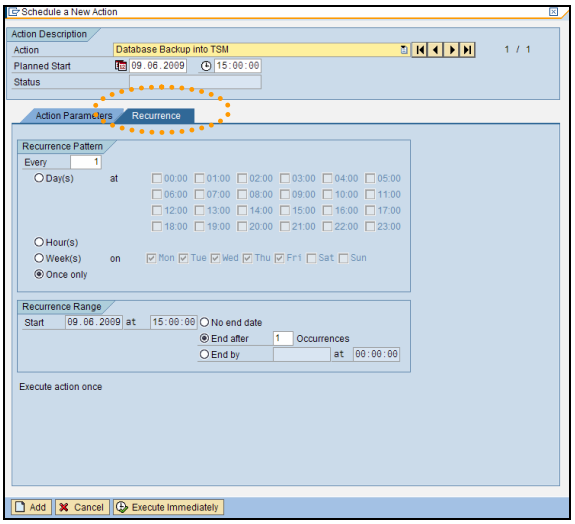


Figure 5-6: Action Dialog Box – Recurrence Settings

You use the recurrence settings to start a database backup that is only running once or at regular intervals. If a backup is supposed to only run once, you can decide to start the backup at a certain predefined date or immediately. To schedule the backup to the point in time in the Planned Start field choose the Add pushbutton. To start the backup action immediately, choose “Execute Immediately”.

The DBA Cockpit allows defining recurrence patterns for a backup as well. For example, we can define to take a database backup every night at 1 a.m. The DBA Cockpit allows scheduling recurring backups based on hours, days and weeks. You can also to define an end date for the schedule. For example, to run a database backup every day for the next 7 days.

Besides manual or scheduled backups based on the DBA planning calendar, the DBA Cockpit allows the configuration of automatic backups. If the database is enabled for roll-forward recovery, both online and offline backups are possible. Online backups will occur during the defined online maintenance window while offline backups are automatically scheduled during the offline maintenance window.

A backup is started automatically based on the following thresholds:

- The time elapsed since the last full backup is more than a specified number of hours
- The number of available DB2 backup images on disk is less than a specified number.
- The time elapsed since the last full backup is more than a specified number of hours.
- The transaction log space consumed since the last backup is more than a specified number

The DBA Cockpit lets you configure the automatic backup based on different pre-configured policies that are optimized for recoverability, performance or a balanced configuration. In addition, you can manually specify the thresholds to trigger a backup. The automatic backup feature can be useful especially for test and development systems where the usage fluctuates over time. It is also useful if

dedicated backup targets are available for a database and there is no need to synchronize backups across the whole IT infrastructure.

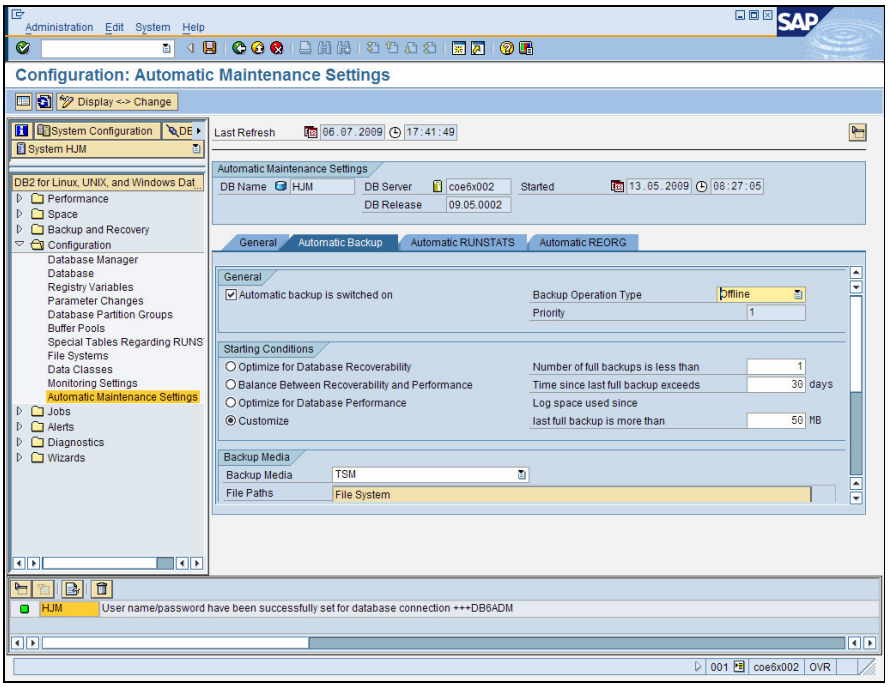


Figure 5-7: DB2 Automatic Backup

To enable automatic backup and to configure the maintenance windows, choose Configuration → Automatic Maintenance Settings in the navigation frame of the DBA Cockpit. The Configuration: Automatic Maintenance screen appears where you can enable and configure automatic backups.

5.3 Monitor Backup History

The DBA Cockpit also lets you monitor the history of database backup operations. The function is similar to the LIST HISTORY command. Therefore, you can monitor the success or failure of any DB2 backup operation – regardless on whether they were scheduled in the DBA Planning Calendar, the DB2 Command Line Processor or other external tools.

5.3.1 DBA Log

As shown in the previous section 5.2, you can schedule database backups using the DBA Planning Calendar. A color scheme is used for the entries to easily determine whether the action completed successfully or not. The corresponding cell in the DBA Planning Calendar is marked red if the corresponding backup job was not successful. If a backup was executed successfully, it is marked green. Backup jobs that are currently running are marked blue. The following figure shows an example of a successful backup job:

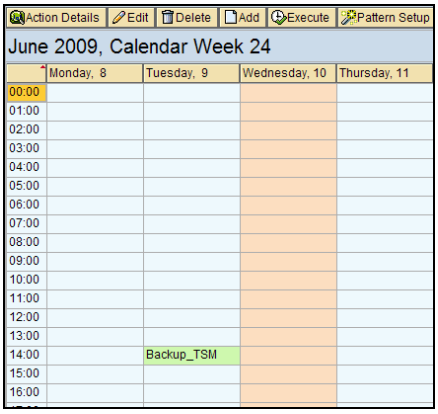


Figure 5-8: DBA Planning Calendar – Job State

For backup jobs that were scheduled via the DBA Planning Calendar, additional information is available in the "DBA log". This log displays the status of the jobs in table format. For each job, the start time, end time, and date are displayed as well as its runtime, the action description and the return code. To access the DBA Log, choose Jobs → DBA Log in the navigation frame of the DBA Cockpit.

The following example shows the entries in the DBA Log that is related to the database backup example above.

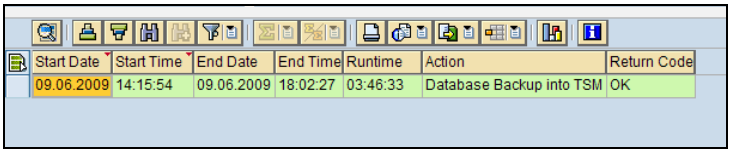


Figure 5-9: DBA Log

In addition to the information that the backup was successful, we see the exact start time and end time of the backup and its runtime. In this example, the scheduled action was a "Database Backup into TSM".

5.3.2 Backup and Recovery Overview – Database Backup

As already discussed, the DBA Log of the DBA Planning Calendar only lists those backups that were scheduled in the DBA Planning Calendar. In addition, the DBA Cockpit is able to provide similar information as the "LIST HISTORY BACKUP" command on DB2 command level. On the Backup and Recovery – Overview screen of the DBA Cockpit, all backup operations are displayed, regardless whether they were triggered in or outside the SAP system.

To access the Backup and Recovery – Overview screen, choose Backup and Recovery → Overview in the navigation frame of the DBA Cockpit. The following Figure 5-10 is an example from the "Backup and Recovery Overview" of the DBA Cockpit:

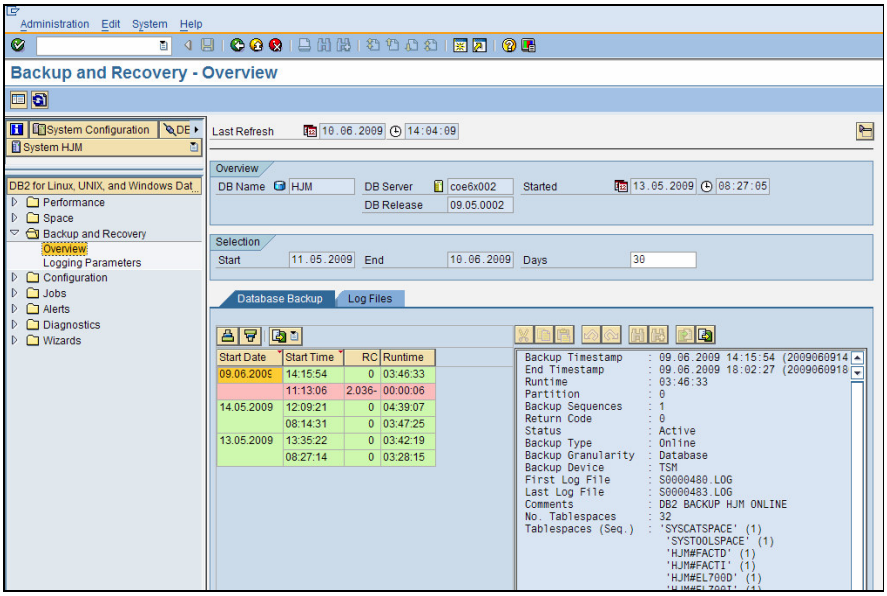


Figure 5-10: Backup and Recovery – Overview – Database Backup

On the Database Backup tab page, a summary of all database backups that were performed in the past is displayed. The screen is divided into two areas. On the left hand side, a list of backups performed is displayed. Similar to the DBA Planning Calendar and the DBA Log, entries are marked green for a successful completion of the backup and red if the backup failed.

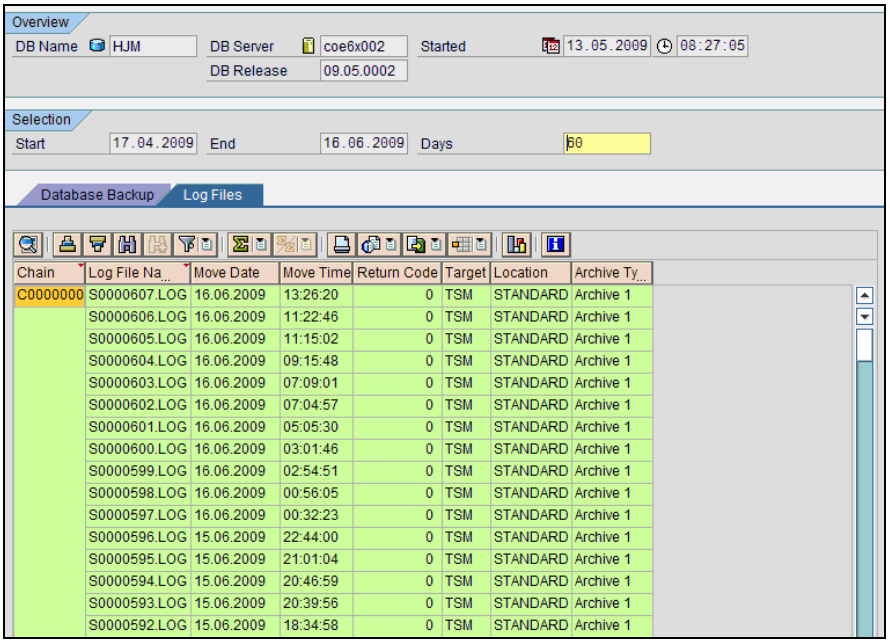
To display its details, choose one entry from this list. The information provided is similar to the output of the LIST HISTORY command. For example, the list of table spaces included in this backup image is shown. Also, information about the log files written during the backup is available.

In Figure 5-10 we see that six backups are listed. While five backups were successful (see the entries marked green) one backup was not executed successfully; see the entry marked red that also has a return code (RC) 2.036-. The return code corresponds to the DB2 return code SQL2036N that in our example means, an invalid device was provided as backup target.

5.3.3 Backup and Recovery Overview – Log Files

The DB2 LIST HISTORY command is able to provide information about database backups as well as the archival of log files. Similar to this, the Backup and Recovery – Overview of the DBA Cockpit offers information about backups and log files.

By switching to the Log Files tabpage of the Backup and Recovery – Overview screen a list of log archive actions is displayed. A green background color implies that log files were successfully archived while a red background color indicates a failure.



Chain	Log File Na...	Move Date	Move Time	Return Code	Target	Location	Archive Ty...
C0000000	S0000607.LOG	16.06.2009	13:26:20	0	TSM	STANDARD	Archive 1
	S0000606.LOG	16.06.2009	11:22:46	0	TSM	STANDARD	Archive 1
	S0000605.LOG	16.06.2009	11:15:02	0	TSM	STANDARD	Archive 1
	S0000604.LOG	16.06.2009	09:15:48	0	TSM	STANDARD	Archive 1
	S0000603.LOG	16.06.2009	07:09:01	0	TSM	STANDARD	Archive 1
	S0000602.LOG	16.06.2009	07:04:57	0	TSM	STANDARD	Archive 1
	S0000601.LOG	16.06.2009	05:05:30	0	TSM	STANDARD	Archive 1
	S0000600.LOG	16.06.2009	03:01:46	0	TSM	STANDARD	Archive 1
	S0000599.LOG	16.06.2009	02:54:51	0	TSM	STANDARD	Archive 1
	S0000598.LOG	16.06.2009	00:56:05	0	TSM	STANDARD	Archive 1
	S0000597.LOG	16.06.2009	00:32:23	0	TSM	STANDARD	Archive 1
	S0000596.LOG	15.06.2009	22:44:00	0	TSM	STANDARD	Archive 1
	S0000595.LOG	15.06.2009	21:01:04	0	TSM	STANDARD	Archive 1
	S0000594.LOG	15.06.2009	20:46:59	0	TSM	STANDARD	Archive 1
	S0000593.LOG	15.06.2009	20:39:56	0	TSM	STANDARD	Archive 1
	S0000592.LOG	15.06.2009	18:34:58	0	TSM	STANDARD	Archive 1

Figure 5-11: Backup and Recovery – Overview – Log Files

The information provided in Backup and Recovery – Overview is very close to the one we discussed in chapter 3 “Log File Management – Details”.

The list in the overview contains information about:

- Chain – Name of the corresponding Log file chain
- Log File Name – Name of the Log file
- Move Date and Time – Date and Time of the Archive Log
- Return Code – “0” means success
- Target – Target Device (in our example always IBM Tivoli Storage Manager)
- Location – Target Location (like Tivoli Storage Manager Management Class)
- Archive Type – Information about the DB2 Archiving method used (in our example “Archive 1” is shown since LOGARCHMETH1 was used to archive the log files)

Copyrights, Trademarks & Disclaimer

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

SAP, SAP NetWeaver, and other SAP products and services mentioned herein are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

More about SAP trademarks at.

http://www.sap.com/company/legal/copyright/trademark.asp

Microsoft, Windows, Windows NT and the Windows logo are registered trademarks of Microsoft Corporation in the United States and/ or other countries.

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

LINUX is a registered trademark of Linus Torvalds.

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

REFERENCES IN THIS PUBLICATION TO IBM PRODUCTS, PROGRAMS, OR SERVICES DO NOT IMPLY THAT IBM INTENDS TO MAKE THESE AVAILABLE IN ALL COUNTRIES IN WHICH IBM OPERATES.

IBM MAY HAVE PATENTS OR PENDING PATENT APPLICATIONS COVERING SUBJECT MATTER IN THIS DOCUMENT.

THE FURNISHING OF THIS DOCUMENT DOES NOT IMPLY GIVING LICENSE TO THESE PATENTS.

INFORMATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND.

THE INFORMATION IN THIS DOCUMENT MAY CONCERN NEW PRODUCTS THAT IBM MAY OR MAY NOT ANNOUNCE.

Any discussion of OEM products is based upon information which has been publicly available and is subject to change. The specification of some of the features described in this presentation may change before the General Availability date of these products.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages.

IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.