



DB2 backup and recovery

IBM Information Management Cloud Computing Center of Competence
IBM Canada Lab

1

© 2011 IBM Corporation

Backing up data is vital for businesses. Loss of information can cause a major crisis or worse, lead to business failure.

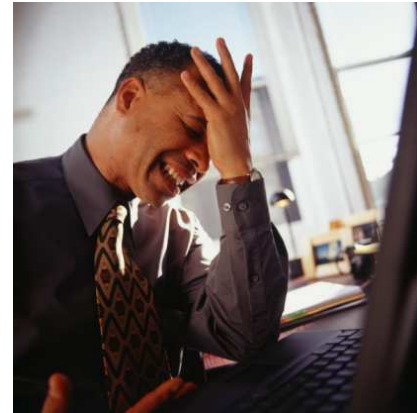
Common problems that can be encountered are:

- A system outage (Power failure, Hardware failure)
- Transaction failure (Users may inadvertently corrupt the database)
- Media failure (Disk drive becomes unusable)
- Disaster (Database facility damaged by fire, flooding or other catastrophe)

DB2 backup and recovery methods are designed to help you keep your information safe!

Agenda

- **Backup and recovery overview**
- **Database logging**
- **Backup**
- **Recovery**



Supporting reading material & videos

- **Reading materials**

- Getting started with DB2 Express-C eBook
 - Chapter 11: Backup and Recovery

- **Videos**

- db2university.com course AA001EN
 - Lesson 10: Backup and Recovery

05/24/2011

3

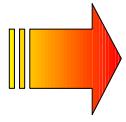
Template Documentation

© 2011 IBM Corporation

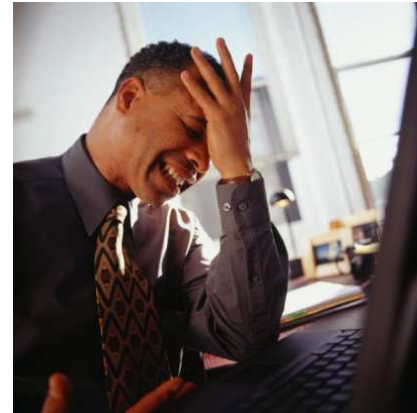
If you need more details about this topic, refer to this supporting material:

- Chapter 11: Backup and Recovery of the book “Getting started with DB2 Express-C 3rd Edition” and
- Lesson 10 of the db2university.com course AA001EN DB2 Academic Training

Agenda

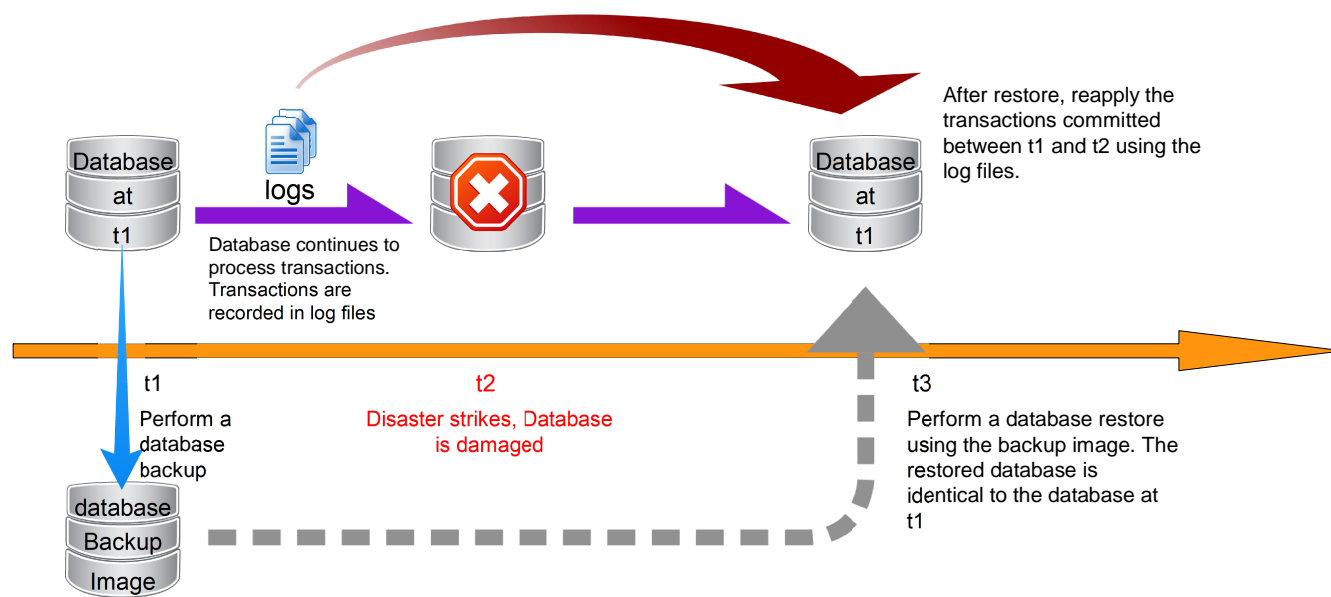


- **Backup and recovery overview**
- Database logging
- Backup
- Recovery



Backup and recovery overview

- At t1, a database backup operation is performed
- At t2, a problem that damages the database occurs
- At t3, all committed data is recovered



05/24/2011

Template Documentation

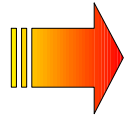
5

© 2011 IBM Corporation

In this figure we briefly talk about backup, restore, and roll forward commands, and the concept of transaction logs.

In the timeline shown, let's say at time t1 you issue the backup command to make a copy of your database. Then users continue working, and the information about their activities is kept at all times in transaction logs. At time t2 there is a problem which damages the database, so at time t3 you issue a restore command using the backup image you took at time t1. This will allow you to get your database back, however, the backup image does not include the information between time t1 and t2, therefore you need to apply the transaction logs for this period. Using the roll forward command you can apply the committed transactions to the restored database, and returned *almost* to the point just before the crash happened. We discuss each of these concepts in more detail in the next slides.

Agenda



- Backup and recovery overview
- **Database logging**
- Backup
- Recovery

Database logging

- Logs keep track of changes made to database objects and their data.
- They are used for recovery: If there is a crash, logs are used to playback/redo committed transactions, and undo uncommitted ones.
- Can be stored in files or raw devices
- Logging is always ON for regular tables in DB2
 - It's possible to mark some tables or columns as NOT LOGGED
 - It's possible to declare and use USER temporary tables which may not be logged

05/24/2011

Template Documentation

7

© 2011 IBM Corporation

If you were working with a text editor, every time you want to ensure your document is saved, you click the save button. In the database world, a COMMIT statement does just that. Every time a COMMIT statement is executed, you guarantee that whatever changes were made to the data, they will be saved somewhere.

In a similar way, when you work with a text document, sometimes you will see at the bottom right corner a brief message saying “auto-saving”. In the database world, this happens as well, because any operation you perform against the data, such as an UPDATE, INSERT or DELETE, will be saved somewhere as you perform it.

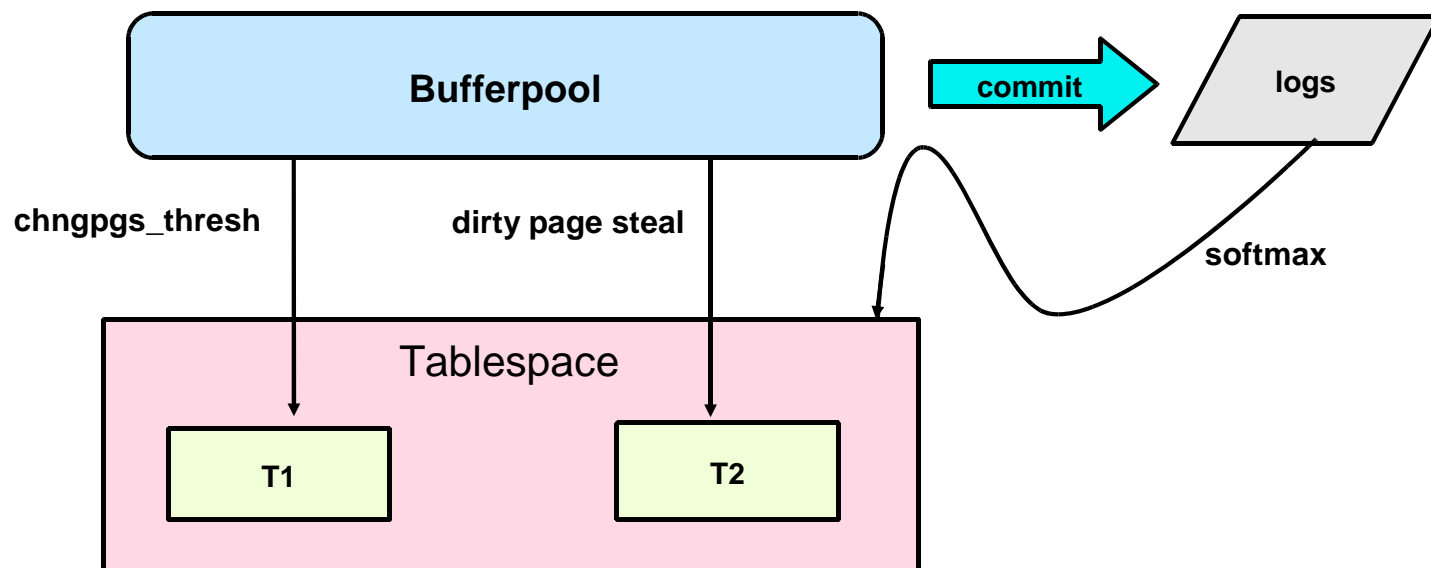
That “somewhere” in the preceding paragraphs refers to the database logs. The database logs are stored on disk and are used to record actions of transactions. If there is a system or database crash, logs are used to playback and redo committed transactions during a recovery.

Database transaction logs keep track of all the operations users perform on a database. For example, the logs will include information about UPDATE statements, including the old values, and the new values. They will have information about INSERT statements with the values being inserted; they will have information about COMMIT or ROLLBACKs, etc.

Logging is ON by default in DB2, and it cannot be turned off. There are some objects or operations that you can choose not to log. For example, you can choose to not log changes to large objects. You can also choose to not log changes to temporary tables.

Database logging

Upon commit, DB2 guarantees data has been written to logs only



05/24/2011

Template Documentation

8

© 2011 IBM Corporation

In the figure we see a table space and logs. Both of them reside on disks, although we recommend that they are not kept on the same disk, because logs are used to recover your data, so if you put both, your data, and the data to recover your data on the same disk, then if the disk fails you lose everything.

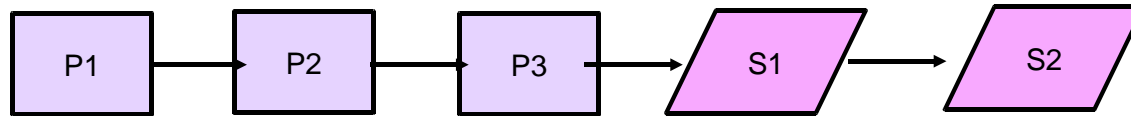
When an UPDATE operation takes place, the pages for the row(s) in question will be brought to the buffer pool (memory). The update changes are performed in the buffer pool, and the old and new values will be stored in the log files, sometimes immediately, and sometimes when a log buffer is full. If a COMMIT is issued after the UPDATE, the old and new value will be stored in the log files immediately. This process is repeated for many other SQL operations that are performed on the database. Only when certain conditions are met, such as reaching the change page threshold specified in the CHNGPGS_THRES parameter, are the pages in the buffer pool “externalized” and written to the table space disk. The CHNGPGS_THRES parameter indicates the percentage of the buffer pool with “dirty” pages, that is, pages containing changes.

From a performance point of view, it does not make sense to perform two writes for each COMMIT operation: One to write to the logs, and another one to write to the table space disk; that’s why “externalization” of the data to the table space disk only occurs when parameters such as the CHNGPGS_THRES threshold are reached.

Types of logs

▪Types of logs – based on log file allocation:

- Primary logs are PRE-ALLOCATED
- Secondary logs are ALLOCATED as needed (costly)



- For day to day operations, ensure that you stay within your primary log allocation

▪Types of logs – based on information stored in logs:

- Active logs:
 - Information has not been externalized (Not on the tablespace disk)
- Archive logs
 - All information externalized

05/24/2011

Template Documentation

9

© 2011 IBM Corporation

There are two types of logs based on file allocation:

Primary logs: These are pre-allocated and the number of primary logs available is determined by the LOGPRIMARY database configuration parameter.

Secondary logs: These are dynamically allocated as needed by DB2. The maximum number of secondary logs is set by the database configuration parameter LOGSECOND. Dynamically allocating a log is costly; therefore, for day to day operations, stay within your primary log allocation. Secondary log files are deleted when all the connections to a database are terminated.

Types of logs based on information stored in the logs:

Active logs

Transactions that have not been committed or rolled back

Online archive logs

Committed and externalized logs in the active log directory

Offline archive logs

Committed and externalized logs in a separate repository

Types of logging

- **Circular logging**
 - For non-production systems
 - Logs that become archived, can be overwritten
 - What if information externalized to the tablespace was wrong? (Human error). No logs to redo things!
- **Archival logging**
 - For Production systems
 - No logs are deleted.
 - Some are stored online (with active logs), others offline in an external media

05/24/2011

10

Template Documentation

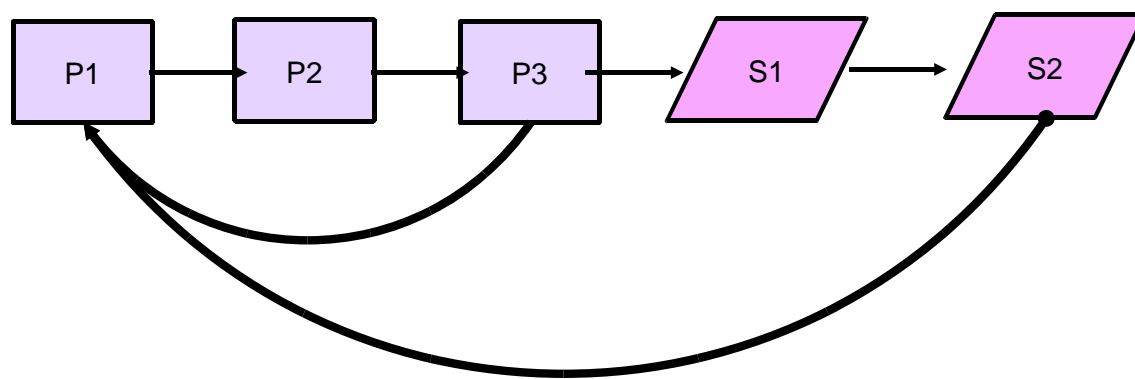
© 2011 IBM Corporation

There are two types of logging:

Circular logging (for non-production systems), were logs whose information has been committed and externalized can be overwritten, and archival logging (for production systems) where logs are never overwritten. Each of these types of logging are described in more detail next.

Circular logging

- Default type of logging
- Logs are overwritten when its contents have been externalized, and there is no need for them for crash recovery
- If a long transaction uses up both, primary and secondary logs, a log full condition occurs and SQL0964C error message is returned
- Cannot have roll-forward recovery



05/24/2011

11

Template Documentation

© 2011 IBM Corporation

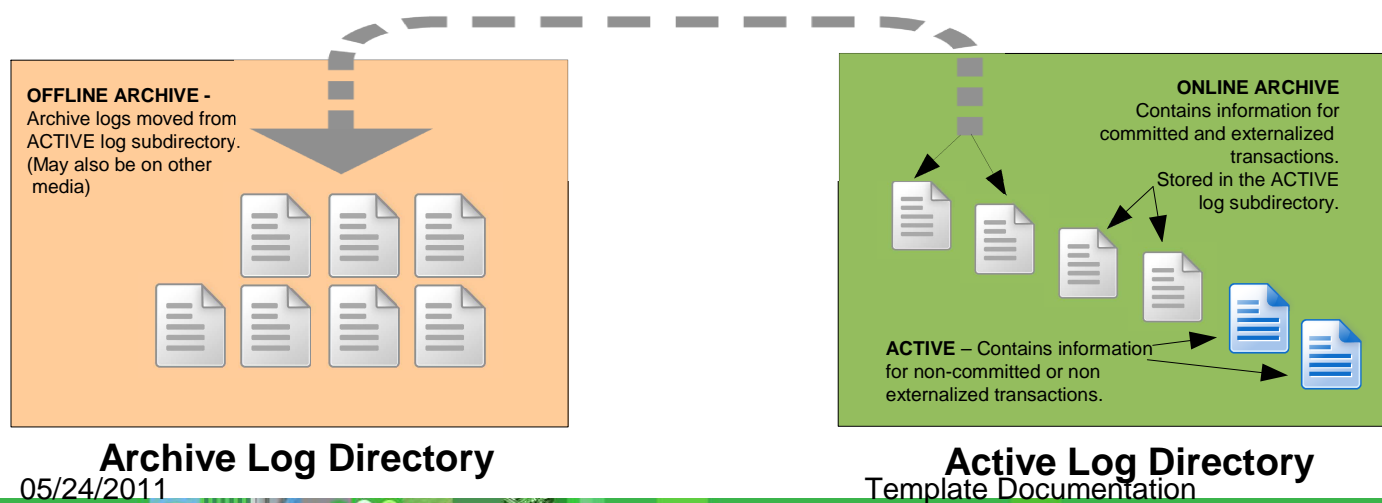
Circular logging is the default, and is enabled when both of the LOGARCHMETH1 and LOGARCHMETH2 database configuration parameters are set to OFF. These parameters indicate the method used to archive the logs, but if you turn them off, that means you do not want to archive the logs, which is how circular logging works. The figure illustrates circular logging:

There are 3 primary logs, therefore we can assume that the value of the LOGPRIMARY parameter is 3. For simplicity, assume there is only one transaction being performed in this example. As the transaction is performed, the log file P1 starts filling up, and then P2. If a commit occurs and the information is later externalized to the table space disk, then P1 and P2 can be overwritten, because the information is no longer needed for crash recovery (which will be discussed in more detail later). If, on the other hand, the transaction is so long that it uses P1, P2, P3, and still needs more log space because the transaction has not been committed nor externalized, then a secondary log (S1 in the figure) is dynamically allocated. If the transaction continues, more secondary logs are allocated until the maximum LOGSECOND logs are allocated. If still more logs are needed, an error message indicating a log full condition is reached will be returned to the user, and the transaction will be rolled back. Alternatively, you can configure DB2 using the BLK_LOG_DSK_FUL configuration parameter to continue writing to the logs every 5 minutes while letting some transactions hang. This gives the DBA some time to find new space, so that the transaction can continue.

Circular logging allows you to recover from crash recovery, but if you want to recover your data to a given point in time, the closest available time would be when you took your last offline backup.

Archival logging

- Enable with LOGARCHMETH1 db cfg parameter
- As soon as enabled, you are asked to take an offline backup
- Log files are NOT deleted – Kept online or offline
- Roll forward recovery and online backup are possible



In archive logging, also known as log retain logging, the log files are not overwritten, but are kept, either online or offline. Online archive logs remain with the active logs which are still needed for crash recovery. Offline archive logs are moved to another media such as tape, and this can be done with USEREXIT routines, Tivoli Storage Manager, or other third party archival products.

To enable archive logging set the database configuration parameters LOGARCHMETH1 or LOGARCHMETH2 (or both) to a value other than OFF. Another way to enable it is to set the LOGRETAIN configuration parameter to RECOVERY. This will automatically cause LOGARCHMETH1 to be set to LOGRETAIN. However, the LOGRETAIN parameter is deprecated and has been left mainly for compatibility with older versions of DB2.

Archive logging is normally used in production systems; because the logs are kept, this allows for database recovery back to point in time as early as the oldest log file. With archive logging, a DBA can recover from errors caused by humans. For example, if a user of a system inadvertently starts performing an incorrect transaction that lasts for days, then when the problem is detected, the DBA can restore the system back to the time before the problem was introduced. However, there may be some manual manipulation required for the transaction to rerun correctly.

Archive logging is required for roll forward recovery and on-line backup. The figure depicts the archive logging process.

Infinite logging

- **Provides infinite active log space**
 - Enabled by turning on archival logging and setting LOGSECOND to -1
- **LOGSECOND indicates how many secondary log files can be allocated. If set to -1, infinite logging allowed**
- **Not good for performance**
 - Secondary logs are constantly allocated
 - Not good for rollback and crash recovery

05/24/2011

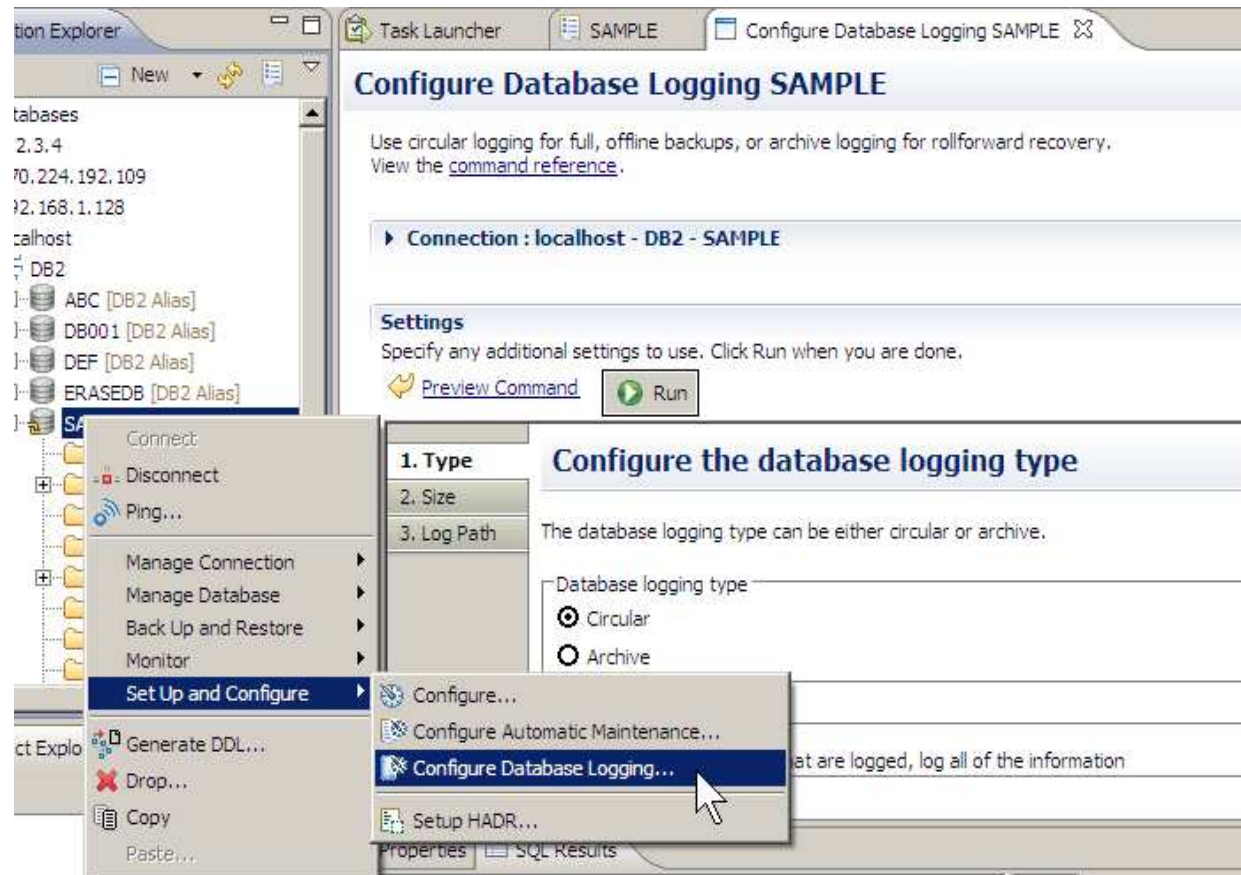
13

Template Documentation

© 2011 IBM Corporation

Infinite logging is possible if you set LOGSECOND to a value of -1; however, this is not recommended as you may run out of file system space, and it's not good for performance in case of crash.

Database logging configuration from Data Studio

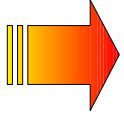


Template Documentation

To set up and configure logging in Data Studio, simply right-click on the database name, and choose “Setup and Configure -> Configure Database Logging”

Agenda

- Backup and recovery overview
- Database logging
- **Backup**
- Recovery



Database backups

- **Copy of a database or table space**

- User data, DB2 catalog, all control files (e.g. buffer pool files, table space file, database configuration file)

- **Backup modes:**

- **Offline Backup**

- Does not allow other applications or processes to access the database
- Only option when using circular logging

- **Online Backup**

- Allows other applications or processes to access the database while the backup is happening

05/24/2011

16

Template Documentation

© 2011 IBM Corporation

The DB2 backup command allows you to take a snapshot copy of your database at the time the command is executed.

Most commands and utilities can be performed online or offline.

Online implies that other users may be connected and performing operations on the database while you execute your command.

Offline means that no other users are connected to the database while you perform your operation. To allow for an online operation, add the keyword **ONLINE** to the command syntax, otherwise, by default the command will be assuming you are executing it offline.

Database backups – Syntax and examples

```
BACKUP DATABASE <dbname>  
[ONLINE] [TO <path>]
```

- Offline backup example

```
backup database sample to C:\backups
```

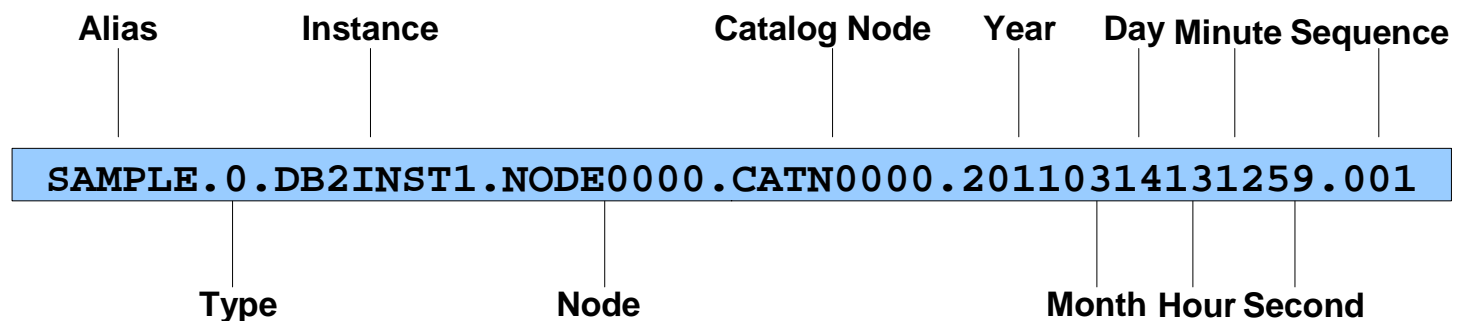
- Online backup example

```
backup database sample online to C:\backups
```

As mentioned earlier, add the keyword ONLINE to the command syntax to make the backup an online operation, otherwise, by default the command will be assuming you are executing it offline.

If you take an ONLINE backup, this means that while users are making changes to the db, you are taking this backup. So the backup image may not contain all the info to the last minute. Therefore it may be good to use the clause "INCLUDE LOGS" to ensure the logs used at the time of the online backup are included with the backup image, this way you have everything should you have to restore from this image.

Database backups – File naming convention



Backup Type:

0 = Full Backup

3 = Tablespace Backup

05/24/2011

18

Template Documentation

© 2011 IBM Corporation

In this slide we have an example of the file name generated for a database backup image. In other words, after you execute the backup command, this is the filename of the file created.

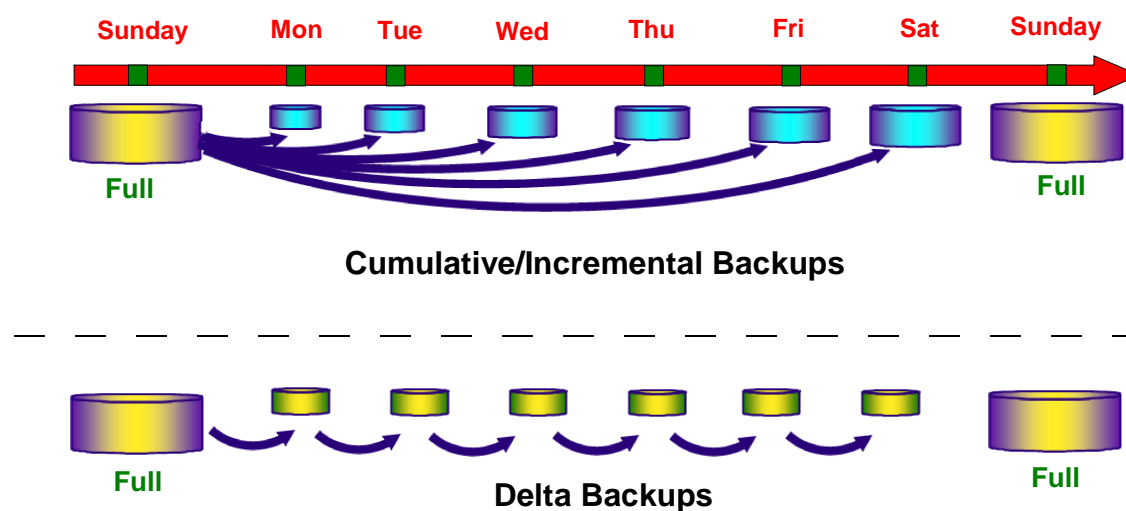
Note that a backup can be performed for the entire database, or just for tablespaces. In the naming convention of backup images, the “type” identifies if this is a database or tablespace backup. If the value is zero, it’s a (full) database backup, if the value is 3 it’s a table space backup.

The “Node” is fixed at NODE0000, and the “Catalog node” is fixed at CATN0000. Only when using DPF (Data Partitioning Feature) will these values change.

The timestamp from the “year” to the “seconds” item are important to remember when you want to restore from a backup copy image as they uniquely identify the backup copy.

Incremental backups

- Suitable for large databases; considerable savings by only backing up incremental/delta changes.



05/24/2011

Template Documentation

19

© 2011 IBM Corporation

Let's now talk about "Incremental backups". Incremental backups can be of two types: Incremental (a.k.a. cumulative) - Backup of all database data that has changed since the most recent successful full backup operation

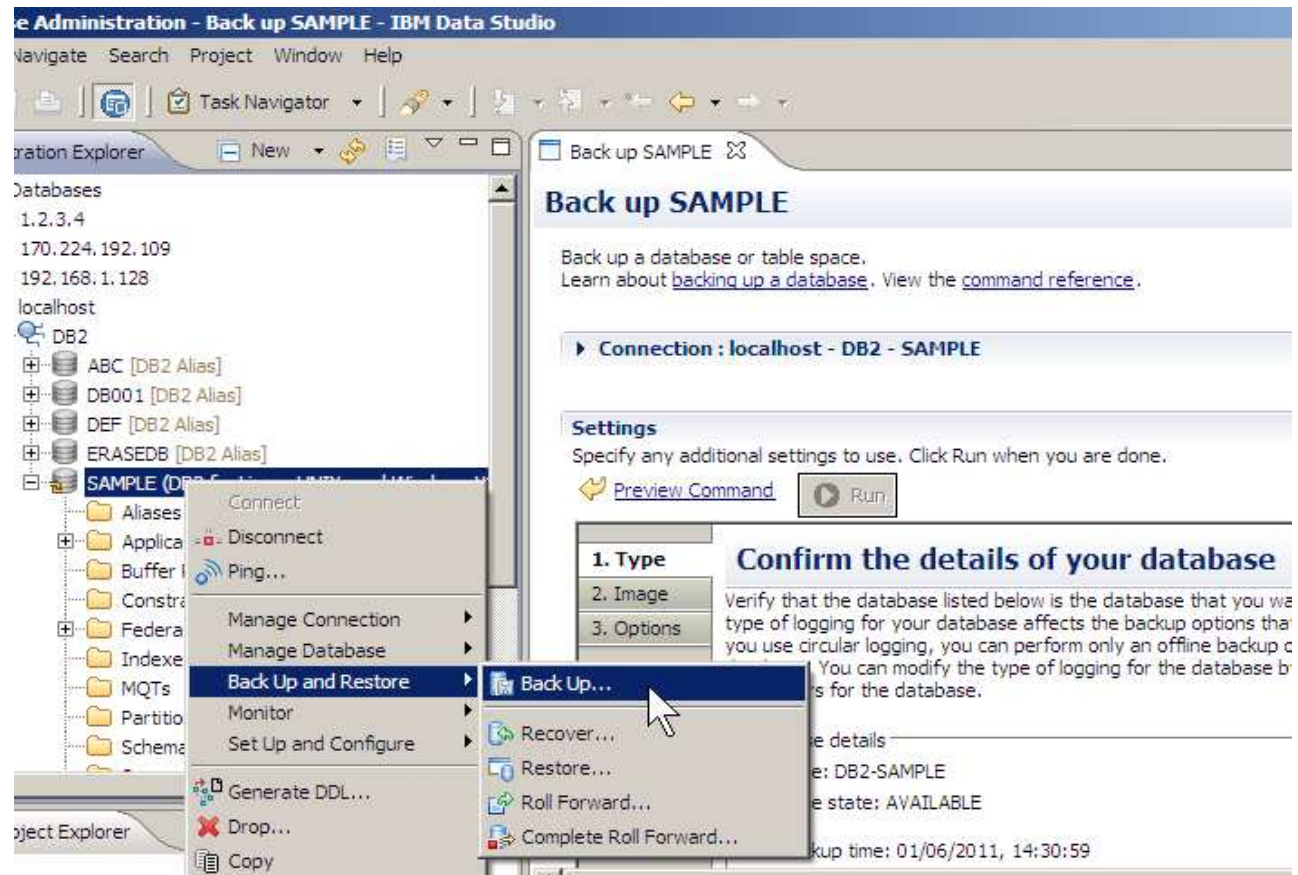
Delta (a.k.a Incremental Delta) - Backup of all database data that has changed since the last successful backup, whether it was a full, incremental, or delta backup operation.

In the figure at the top, you take a full backup on Sunday, then on Monday you take a cumulative backup which will include all changes from Sunday to Monday. Then on Tuesday, you take another cumulative backup. This backup image will include all the changes from Sunday to Tuesday (it's cumulative, so it will also include the changes from Sunday to Monday), and so on. In this example, if there was a crash prior to taking a cumulative backup on Friday, then to restore to the closest point in time, you would have to restore from the full backup first taken on Sunday, and then on top restore the cumulative backup taken on Thursday.

At the bottom of the figure or delta backups: Say you take a full back on Sunday. Then on Monday you take a delta backup with changes from Sunday to Monday. Then on Tuesday, you take another delta backup, which will store the changes only from Monday to Tuesday, and so on, and so on. So in this example, if there was a crash prior to taking the delta backup on Friday, then to restore to the closest point in time, you would have to restore from the full backup first taken on Sunday, and then on top restore each of the delta backups taken on Monday, Tuesday, Wednesday and Thursday.

Fortunately DB2 keeps a history of the different types of backups you have taken and can intelligently determine which images should be used to restore your database. For incremental backup to work, you need to have the TRACKMOD database configuration parameter set to ON.

Database backups from Data Studio



05/24/2011

Template Documentation

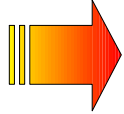
20

© 2011 IBM Corporation

From Data Studio, it's easy to perform a backup. Simply right-click on the database name, choose Back Up and Restore, and then "Back up"

Agenda

- Backup and recovery overview
- Database logging
- Backup



- **Recovery**

Database recovery

- A database recovery will recreate a database or tablespace from backups and logs
- Use the restore and rollforward commands



05/24/2011

Template Documentation

22

© 2011 IBM Corporation

A database recovery implies restoring your database from a backup and/or logs. If you just restore from a backup, you would be recreating the database as it existed at the time the backup was taken.

If archive logging was enabled before the backup, you can not only restore using a backup image, but also from the logs. As we will see in the next section, a roll-forward recovery allows you to restore from a backup, and then apply (roll-forward) the logs to the end of the logs, or to a specific point in time.

Note that the term “recovery” is used often in this section, but the command used for recovery is called RESTORE.

Types of database recovery

■ Crash recovery

- Protects the database from being left inconsistent (power failure)

■ Version recovery

- Restores the database from a backup.
- The database will return to the state as saved in the backup
- Any changes made after the backup will be lost

■ Rollforward recovery

- Needs archival logging to be enabled
- Goes through the logs to reapply changes on top of the backup.
- It is possible to roll forward either to the end of the logs or to a specific point in time.
- Minimal data loss

05/24/2011

23

Template Documentation

© 2011 IBM Corporation

Crash or restart recovery

Assume you are working on a desktop computer running important transactions to a DB2 database. Suddenly there is a power outage, or someone accidentally unplugs the power cord: what will happen to the database?

The next time you start your computer, and start DB2, crash recovery will automatically be executed. In crash recovery, DB2 will automatically run the command `RESTART DATABASE` and will read and redo/undo the transactions based on the active logs. When this command completes, you are guaranteed that your database will be in a consistent state, that is, whatever was committed will be saved, and whatever was not committed will be rolled back.

Version or image recovery

This type of recovery implies that you are restoring only from a backup image; therefore, your database would be put in the state it was at the time the backup was taken. Any transactions performed on the database after the backup was taken would be lost.

Roll-forward recovery

With this type of recovery, you not only `RESTORE` from a backup image, but you also run the `ROLLFORWARD` command to apply the logs on top of the backup so that you can recover to a specified point in time. This type of recovery minimizes data loss.

Database recovery – Restore command syntax

```
RESTORE DATABASE <dbname>  
[ONLINE][FROM <path>]  
[TAKEN AT <timestamp>]  
[WITHOUT PROMPTING]
```

▪ Offline restore example

SAMPLE.0.DB2INST.NODE0000.CATN0000.**20110718131210**.001

```
restore database sample  
from C:\backups  
taken at 20110718131210
```

05/24/2011

24

Template Documentation

© 2011 IBM Corporation

This is the syntax of the RESTORE command.

Use the RESTORE command to recover a database from a backup image. In the syntax, “Online” is needed to perform this operation online, which means users are connected to the database. Online restore would only apply to a table space restore.

“Taken At” is where you specify the timestamp that you get from the backup copy image.

“Without prompting” is used so there are no prompts during the restore operation. For example, say you restore over an existing database. The RESTORE command would prompt you if you want to overwrite the existing database or not. If you use “Without prompting” the restore command would just go ahead and overwrite it.

At the bottom of the slide there is an example of restoring a database. Note that TAKEN AT clause is used to specify from which backup image to restore. As seen on the example and explained earlier, the name of the backup image includes the timestamp when it was taken.

Table space backups & restores

- Enables user to backup a subset of database
- Can only be used if archival logging is enabled
- Multiple table spaces can be specified
- Table space can be restored from either a database backup or table space backup
- Use the keyword **TABLESPACE** to specify table spaces
- Example: Backing up online a tablespace 'tblsp1'

```
backup database sample
    tablespace(tblsp1)online to C:\backups

restore database sample
    tablespace(tblsp1)online from C:\backups
without prompting
```

05/24/2011

Template Documentation

25

© 2011 IBM Corporation

This slide shows that both, backup and restore operations can be performed at the table space level.

Note that when you restore and roll forward a table space, you have to do it to a minimal point in time (PIT) which ensures your data is consistent. To find out the minimal PIT, use the command: `GET SNAPSHOT FOR TABLESPACES ON <database>`

and look for “Minimum Recovery Time” item.

You can also use the “list tablespaces show detail” but an item for minimum point in time recovery won't appear unless archival logging is enabled.

Multiple table spaces can be specified when backing up/ restoring. Ensure to use the keyword **TABLESPACE** followed by the table space names in the command.

Note as well that table spaces can be restored from either a database backup or a table space backup.

At the bottom of the slide there is an example of backing up a table space, and then restoring it, both performed online.

Also possible with backup and restore...

- Backup compression
- Clone a database from a backup image and change containers (redirected restore)
- Restore over existing database
- Recovery of dropped tables
- etc...

05/24/2011

26

Template Documentation

© 2011 IBM Corporation

This final slide shows other things that are possible with backups and restores.

Backup images can be compressed which can save space.

Cloning a database can be performed with a backup/restore operations; however if you take a backup on Linux, it cannot be restored on Windows and viceversa. In some cases within UNIX, it is possible to backup a database taken on one UNIX, and restore it on another UNIX. For cases where you need to clone a database from one Windows system to another one on Linux (or viceversa) you need to use db2look and db2move utilities.

In addition, what happens if you want to backup and restore a database on systems running the same operating system, but with different disk layout? For example if you take a backup on a system with 5 disks, and want to restore it on another system that has 3 disks? The backup image includes information about where the table spaces are stored. If the system you want to restore to has a different disk layout, you need to run a redirected restore. Using a redirected restore, you can manually (or using a script) specify the containers for your table spaces.

You can restore over an existing database, though you will receive a warning message.

Recovery of dropped tables is possible if you turn on this support which would cause more logging and performance cost.

Note that in addition to restore and roll forward, there is another command called "Recover" which performs a "Restore" plus a "Roll forward" in one single command.



Thank you!

Use the forum in the db2university.com course AA001EN if you have technical questions about the materials covered in this course. Fellow students, faculty and IBMers can help you!