DBA Certification Course

(Summer 2008)

Chapter 6: Backup and Recovery

- Log Files
- Backup
- Recovery

Objectives

After completing this chapter, you should be able to:

- explain the role of log files
- develop a recovery strategy
- perform a backup
- recover a database after media failures

Literature (1)

- Hana Curtis: DB2 9 Fundamentals exam 730 prep, Part 5: Working with DB2 objects
  [http://www.ibm.com/developerworks/edu/dm-dw-db2-cert7305.html]
- Clara Liu, Raul Chong, Dwaine Snow, Sylvia Qi: Understanding DB2: Learning Visually with Examples
  IBM Press/Pearson, 2005, ISBN 0-13-185916-1, 895 pages.
- DB2 for Linux, UNIX, and Windows Version 9 Information Center
  [http://publib.boulder.ibm.com/infocenter/db2luw/v9//index.jsp]

Overview

1. Motivation, General Remarks
2. Logging
3. Backup and Restore
4. Rollforward, Restore
5. Advanced Topics

Introduction (1)

- It can be very expensive when data are lost and must be re-entered manually.
- It can be even more problematic when the data cannot be re-entered anymore.

  E.g. consider orders, shipments, payments.
- There are also legal requirements that data about business transactions must be kept for a long period of time (10 years).
- It is also bad for the business if the database is not available for a longer period.

Introduction (2)

- In order to plan methods for protecting the data, one must think about possible risks:
  - ◇ a disk is damaged (media failure),
  - ◇ when repairing a RAID system, the disk IDs are mixed up, thus sectors are written to the wrong disks, all data is destroyed,
  - ◇ an administrator accidentially drops an important table,
  - ◇ a fire in the computer room destroys all equipment (and the backup tapes in this room).

Introduction (3)

- It is important to check that the backup copies are indeed readable and contain the necessary data.

  One does not want to find out that the backup script did not work only after the disk does not work anymore.
- One should develop a strategy for backup and recovery, write it down, and practice it from time to time (on a test system).
- Notice also that with the default settings, Oracle and DB2 have no protection against media failures.

  One must explicitly change settings to ensure that log files are archived, do backups from time to time, delete unnecessary log files.

Introduction (4)

- It is important that the administrators know what to do when recovery is needed.
  - ◇ Everybody is excited and wants the system back running as fast as possible.

    Every minute of downtime costs the company money.
  - ◇ But if the administrator does the wrong things, he/she might make the situation worse.
  - ◇ There is also no single method for recovery: The fastest way to get the system running again depends on what exactly was damaged.

Overview

1. Motivation, General Remarks
2. Logging
3. Backup and Restore
4. Rollforward, Restore
5. Advanced Topics

Logging (1)

- Suppose that a transaction does updates, i.e. it changes database pages in the buffer.
- The changes are written to the log buffer (information to undo and to redo the change).
- After a short time, the log buffer is written to disk (to the log files). This must be done before the commit can complete.
- Only after the log information was written, the changed database pages can be written to disk (WAL principle: write-ahead logging).

Logging (2)

- The writing of changed database pages can be done asynchronously, and possibly with a quite long delay after the `COMMIT`.
- No information can be lost, since the change is recorded in the log files.
- If the contents of main memory should be lost (e.g. because of a power failure), the log files can be scanned and the changes redone.

  Typically, the page header contains the number of last applied log entry. Therefore, when the DBMS restarts after the crash, it can see whether a change was already applied to the page version on disk.

Logging (3)

- Advantages of delayed writing of database pages:
  - ◇ A transaction can change many pages, but the log entries are typically much smaller and might fit into a single page.
  - ◇ If the log files are on a separate disk, the disk head is often already on the right cylinder.
  - ◇ A page might be changed many times before it is written.
  - ◇ If many pages have to be written asynchronously, they can be ordered by their position on disk.

Logging (4)

- In DB2, the writing of changed ("dirty") database pages is done by background processes called "page cleaners" (`db2pclnr`).

  The db cfg parameter `NUM_IOCLEANERS` determines the number of these processes.

- Pages successfully written to disk are called "externalized".
- There are also processes for asynchronous reading, the buffer pool prefetchers (`db2pfchr`).

  Their number is determined by the db cfg parameter `NUM_IOSERVERS`.

Logging (5)

- The log buffer is written to the log files by a background process called "DB log writer" (`db2loggw`).
- If the log needs to be read (for transaction rollback or during recovery) this is done by the "database log reader" (`db2loggr`).

  The process `db2logts` writes the file `DB2TSCHG.HIS` (table space change history), which is used to determine which log files are needed for recovery of a particular table space.

- All these processes exist once per database.

  Under Windows, they are threads. IBM calls them EDUs: DB2 Engine Dispatchable Units.

Logging (6)

- The parameter `logbufsz` determines the size of the log buffer.

  The memory is taken from the database heap.

- The parameter `mincommit` can be used to delay the writing of the log buffer for a `commit` a bit, which might increase the overall throughput (then it is written only once for a group of transactions).

  `mincommit` is the minimum number of transactions that must issue a `commit` command before the log buffer is written. However, the delay is at most one second. This is important because no `commit` can succeed before the changes done by the transaction are safely stored in the log files. Thus, the application has to wait.

Log Files (1)

- The log files are initially created in the directory `SQLOGDIR`, which is a subdirectory of the database directory.
- The log files are called `S0000000.LOG` and so on.
- The directory can be changed by assigning a value to the db cfg parameter `newlogpath`.

  The current path is displayed with `GET DB CFG FOR` ⟨`DB-Name`⟩.

- The parameter `logfilsiz` specifies the size of each log file in 4K pages.

Log Files (2)

- The parameter `logprimary` determines the number of primary log files.

  Primary log files are created when the database is activated (either when the first application connect to the database, or when an `ACTIVATE DATABASE` command is issued.

- A log file is called active if it contains log entries that might be needed for crash recovery, i.e.
  - ◇ changes by transactions that are not yet finished,
  - ◇ changes by commited transactions that are not yet externalized.

Log Files (3)

- In the default logging method (circular logging), inactive log files are reused (overwritten).
- If all primary log files are active, and more log space is needed, secondary logfiles are created.
- The maximum number of secondary logfiles is determined by the parameter `logsecond`.
- If this space is also used up, and all logfiles are still active, a "log full" error occurs and the transaction is rolled back.

Log Files (4)

- It is very important not to lose log files:
  - ◇ The last changes are documented only in the log files, so they would be lost.
  - ◇ While the database is running, the database files are usually not in a consistent state: some changed pages are already written back to disk, others are not (because they were accessed again by another transaction).
- Therefore, log files are usually mirrored (stored on two different disks): Parameter `mirrorlogpath`.

Crash Recovery (1)

- In case of a power failure or software "crash", the contents of main memory is lost.
- In this case, only the active log files are needed:
  - ◇ First, the log files are scanned in forward direction, and all changes (that did not make it to disk) are redone.
  - ◇ Furthermore, the set of uncommitted transactions is determined in this phase.
  - ◇ Then the changes by the uncommitted transactions are undone in the opposite direction.

Crash Recovery (2)

- Crash recovery is done automatically at the next startup when the database detects that it was not correctly shut down.

  DB2 has a parameter `autorestart` which determines whether crash recovery is done automatically. It is `on` be default. If it is `off`, an administrator must explicitly say "`RESTART DATABASE`".

- Without crash recovery, the database files are in an inconsistent state after the abnormal termination, and are therefore unusable.

Version Recovery

- When circular logging is used, the database is non-recoverable. Then only the following is possible:
  - ◇ Crash recovery (see above).
  - ◇ "Version recovery" means that in case of a media failure (disk failure), an old backup can be restored, so one gets an old version of the DB.

    All changes since the time of the backup are lost.
- Furthermore, backups for non-recoverable databases can only be done offline.

  I.e. no application can access the database while the backup is taken (in order to get a consistent state).

Rollforward Recovery (1)

- Rollforward recovery means to
  - ◇ restore a backup, and then
  - ◇ apply the changes in all log files since the backup was made ("roll forward the database").
- Thus, log files must be kept (in an archive), even after they became inactive.

  Inactive log files are no longer needed for crash recovery. They are also called archive logs.

- A database in which the necessary log files are available is called a recoverable database.

Rollforward Recovery (2)

- In a recoverable database, backups can be done
  - ◇ offline (when the database is shut down), or
  - ◇ online (while the database is being used).

    Since the database files change while the backup copy is being made, one needs all log files that were active at the beginning of the backup, and all log files produced during the backup to be able to restore a consistent database state.
- With a recoverable database, one can back up, restore, and roll forward also individual tablespaces, not only the entire database.

  And the table space restore can be done while the rest of the database is online.

Log Archival (1)

- DB2 has parameters `logarchmeth1` and `logarchmeth2` that specify how log files can be archived:
  - ◇ `OFF`: circular logging (if both are `OFF`).
  - ◇ `LOGRETAIN`: log files remain in the log directory.
  - ◇ `DISK:`⟨`Path`⟩: log files are copied to specified dir.
  - ◇ `USEREXIT`: The program `db2uext2` is called for each log file that should be archived.

    IBM supplies example source code.
  - ◇ `TSM:`⟨`Param`⟩: Tivoli storage manager is called.
  - ◇ `VENDOR:`⟨`Library`⟩: A vendor library is called.

Log Archival (2)

- If both archival methods are used, one gets two copies of the log files.
- It happens sometimes that an archival method fails, e.g. because the tape or disk is full. One can specify a failover archive path with `failarchpath`.

  This directory is a temporary storage for log files that could not be archived because the archival method failed. If the archival method becomes available again, the files are archived and deleted from this directory.

  The parameter `archretrydelay` determines the number of seconds to wait before the next call to the archival method after a call failed. There are `numarchretry` attempts before a file is saved in the failover archive path.

Log Archival (3)

- After a log file was archived and became inactive, it can be reused.

  DB2 still uses increasing file names, but simply renames an existing file (which is then overwritten).
- If files cannot be reused (e.g. because `LOGRETAIN` was selected as archival method), the log disk can become full.

  By default, a transaction is rolled back if it cannot write to the log. The database might even come down under certain conditions. If one sets `blk_log_dsk_ful` to `YES`, the transaction simply waits until the log can be written again. DB2 tries this every five minutes and writes a message to the administration notification log if it fails.

Infinite Active Logging

- When all possible primary and secondary log files are still active, and attempt to create more log files fails and the transaction is rolled back.
- If some transactions run for a long time or produce a large amount of log data, there can be problems.
- When log files are archived, one set `logsecond` to `-1`, meaning that there is no limit on the number of active log files: If necessary, log files are fetched back from the archive.

Overview

1. Motivation, General Remarks
2. Logging
3. Backup and Restore
4. Rollforward, Restore
5. Advanced Topics

Backup (1)

- A database backup contains:
  - ◇ All database objects (i.e. all data files).
  - ◇ Information about table spaces and containers.
  - ◇ The database configuration file.
  - ◇ The log control file.
  - ◇ The recovery history file.
- It does not (!) contain:
  - ◇ The database manager configuration file.
  - ◇ Values of DB2 registry variables.
  - ◇ Application programs etc.

Backup (2)

- Example for simple offline backup command:

```
BACKUP DATABASE sample TO e:\backups
```

- Required authority: `SYSADM`, `SYSCTRL`, or `SYSMAINT`.

  Note that `DBADM` cannot perform a backup.

- The command automatically opens a connection to the specified database, therefore it might be necessary to specify user and password:

```
BACKUP DATABASE sample USER brass USING mypw
        USE TSM
```

  Here, the backup data are stored on the Tivoli Storage Manager.

Backup (3)

- The default backup method is an offline backup.
- For recoverable DBs, online backup is possible:

```
BACKUP DATABASE sample ONLINE TO d:\backups
        INCLUDE LOGS
```

- The optional clause `INCLUDE LOGS` means that the log files that are required to make the backup copy consistent are included with the backup.

  Of course, log files are anyway archived, so this might not be necessary. However, if the log files should somehow be lost, an online backup is worthless: Not even version recovery is possible. Therefore, it might be good to package the necessary log files with the backup.

Backup (4)

- One can also backup specific tablespaces:

```
BACKUP DATABASE sample
       TABLESPACE (syscatspace, usersp1, usersp2)
       ONLINE
       TO d:\backups
       INCLUDE LOGS
```

- One should backup related tablespaces together.

  E.g. if table data, indexes, and lob data of a table are stored in different tablespaces, they should all be included in one backup. Also tables with foreign key constraints between them are good candidates for a common backup.

Backup (5)

- To reduce storage requirements, and the time needed for the backup, one can also do incremental backups.
- In order to use incremental backups, one must first set the db cfg parameter `trackmod` ("track database modifications") to `YES`.

  After this parameter was set to `YES`, a full backup must be taken.

- Then the backup program does not have to scan all pages to find out which were changed since the last backup.

Backup (6)

- An incremental cumulative backup contains all pages that were changed since the last full (normal) backup:

```
BACKUP DATABASE sample ONLINE INCREMENTAL
       TO d:\backups
```

- An incremental delta backup contains all pages that were changed since the last full, cumulative, or delta backup:

```
BACKUP DATABASE sample ONLINE INCREMENTAL DELTA
       TO d:\backups
```


Backup (7)

- Example backup schedule:
  - ◇ Full backup on Sunday,

    In this example, all backups are done in the late evening/night.

  - ◇ delta backups on Monday, Tuesday, Thursday, Friday,
  - ◇ cumulative backup on Wednesday.
- If something happens on Friday, one must restore
  - ◇ the full backup from last Sunday,
  - ◇ then the cumulative backup from Wednesday,
  - ◇ and then the delta backup from Thursday.

Backup (8)

- The names of backup files have the following components (subdirectories or concatenated, separated with "."):
  - ◇ Database alias (e.g. `SAMPLE`)
  - ◇ Type of backup (e.g., 0=Full, 3=Table Space)
  - ◇ Instance name (e.g. `DB2`)
  - ◇ DB partition (`NODE0000` for single partition DB)
  - ◇ Catalog partition (`CATN0000` for single part.)
  - ◇ Timestamp of the backup (`YYYYMMDD HHMISS`)
  - ◇ Image sq. number (one backup in several files)

Restore (1)

- Simple restore command:

```
RESTORE DATABASE sample
        FROM d:\backups
        TAKEN AT 20080920050000
        WITHOUT ROLLING FORWARD
        WITHOUT PROMPTING
```

- "`WITHOUT ROLLING FORWARD`" means that one wants the version from the backup.

  Otherwise, the database is put in "roll forward pending" state, which means that one must do next the roll forward. For an online backup, the roll forward pending state is always set.