## Chapter 6.5 Protection on Medium Failure

*Definition:*

*Logging can provide Data Protection which aimed with System Malfunction, when System Malfunction happens, then there will lose nothing in the Disk while temporary memory would lost its data. The much more serious malfunction would have lost one or several Disks.*

Below, we would discuss the Backup System that although the data lost in the Disk and it would not influence the Recovery of Database.

Chapter 6.5.1 Backup

*Definition:*

In order to provide Data Protection for Medium Failure, we need to use the technique which is called *Archiving Solution which is used to maintain and separate the Database Copy from Database itself.*

However, if there is any possibility to shut down the Database temporarily, then we can use *Storage Medium to create the Backup File, and copy Database to a safety place far away. With Backup File, if there happens Medium Failure, then Database System would be recovered to it’s previous status.*

If we want to proceed to a much closer procedure, then we can use Logging File, and precondition is that the Backup Logging can be saved and after any Failure, the Logging File itself still exists. *In order to prevent the lost of Logging, we need to send the copy of Logging to a far away site.* If Logging and Data are all lost, then we can use the Backup Logging to recover, at least to recover to the time when the Logging was last sent.

*Building Backup is a long period, so we try to avoid copying the whole Database System during each step when building backup. Here, Two types of Backup:*

1. *Completely Unloading - Copy the whole Database.*
2. *Increment Unloading - Copy those changed Database Elements after the Completely Unloading or Increment Unloading. There have several levels Unloading, and Completely Unloading is considered as the ‘0 Level’ Unloading, ‘i Level’ Unloading is all changes that are less than or equal to ‘i Level’ Unloading.*

Here, we can use one Completely Unloading and its sequential Increment Unloading to recover the Database, and the process is totally the same as using Redo Logging File or Undo/Redo Logging File to repair System Failure.

* *Use Completely Unloading to recover the Database at first.*
* *According to time - line in the Increment Unloading, make changes on the current new Database by using Increment Unloading.*

Chapter 6.5.2 Non - Static Storage

*Definition:*

Most of the Database System can not be closed during backup and copy which may needs some period of time. Therefore we choose the Non - Static Storage which is similar with the Non - Static Checkpoint.

*Non - Static Storage:*

* Non - Static Storage tries to build the uploading the copy of Database at the most start, however, during several minutes when proceeding Uploading, Database Activity may need to change many Database Elements. If we need to recover Database from Backup File, during the Process of Uploading, the Logging File can be used to arrange the Data here, and tries to make the Database System consistent.
* There also has another possibility that when Uploading, some Operations may change Database Value, at that time, Backup System may looks totally different from Current Database System. As long as Logging File Records are saved in the Disk, then we can utilize Logging File Record and repair Database System, which is to say to make it consistent in Database System.

*Example:*

Give a simple example, assume that the Database system consists of four basic elements, which are A, B, C, and D, initially, they are set as 1, 2, 3, and 4. During the process of Uploading, A turns to 5, C turns to 6, B turns to 7. Although at the very start, the database values are (1, 2, 3, 4), but when it starts to uploading, then the database values turn to (5, 7, 6, 4), but in the Backup Database, the database value turns to (1, 2, 6, 4), so during Uploading, the Database Status is the inconsistent status.

|  |  |
| --- | --- |
| Disk | Copy |
|  | Copy A |
| A := 5 |  |
|  | Copy B |
| C := 6 |  |
|  | Copy C |
| B := 7 |  |
|  | Copy D |

More precisely, the process to build backup file can be divided into several steps, we assume that the type of Logging File is Redo or Undo/Redo Logging. Also Undo Logging is not appropriate to be used with Backup.

* *Write Logging Record <START DUMP>.*
* *Execute the appropriate Checkpoint according to the appropriate Logging Record Type.*
* *Execute the completely Uploading or Increment Uploading based on the need, make sure that the Data copy has reached the Remote Node.*
* *Make sure that enough Logging File has been copied to the safety Remote Node, at least to ensure the 2nd one that the checkpoint and former checkpoint do exist on the Disk after the Medium Malfunction.*
* *Write into Logging Record <END DUMP>.*

After Uploading finished, then we can make sure that the Logging that starts before the checkpoint before the 2nd one is safe. Which is to say that eliminating all Logging Record before 2nd Checkpoint is safety.

*Example:*

Assume that the changes that the Transactions make on the Database are caused by Transaction T1 (Write A and B.) and Transaction T2 (Write C.), but they are active when Uploading starts. The image gives the possible Undo/Redo Logging File during the Transaction. Attention that, Transaction T1 has not been committed.

|  |
| --- |
| *<START DUMP>* |
| *<START CKPT (T1, T2)>* |
| *<T1, A, 1, 5>* |
| *<T2, C, 3, 6>* |
| *<COMMIT T2>* |
| *<T1, B, 2, 7>* |
| *<END CKPT>* |
| *Dump completes* |
| *<END DUMP>* |

Chapter 6.5.3 Recovery by using Backup and Logging