## Chapter 6.4 Undo/Redo Logging File

*Background:*

In this Chapter, we already saw two different Logging Schema, the difference here is just that when adding Modification Logging Record, *Undo Schema is using the OLD value while Redo Schema is using the New value*. Of course, each one has it’s own defect:

* ***Undo Logging*** - Write the data back right after the end of Transaction which has not been updated to the Disk, and this increases the necessary Disk I/O.
* ***Redo Logging*** - Before we commit and flush the Logging Record, we need to keep any modifications of the Blocks into Buffer Area, which may increase the average Buffer Areas.
* Database Element is not integrity block or block collection, there will be ***Contradiction*** for Undo Logging and Redo Logging during the process of Checkpoint.
  + *There has one buffer area which includes database element A that has been modified by committed Transaction.*
  + *There is another situation that in the same Buffer Area, exists another Database Element B which has been modified by UNCOMMITTED Logging Record.*
  + *Analysis from the first situation, Database Changes should be copied to the Disk. While the second situation, Database Changes should not be copied into the Disk according to the RU1 which requires that the Transaction Logging Record should be updated to the Disk and after that, Database Changes can be updated to the Disk.*

In this Chapter, we would look *Logging Type of undo/redo*, which can be used to maintain much more information (which is still one type of cost.), provide much more flexibility on the motion sequence.

Chapter 6.4.1 Undo/Redo Rule

Chapter 6.4.2 Recover by using Undo/Redo Logging File

Chapter 6.4.3 Checkpoint in Undo/Redo Logging File