## Chapter 4.7 Buffer Management

Assume that the operator on Relation can be used to get M main memory buffer, and of course they can be used to store the required data. *Provide available main memory for Query on Database, this key task is assigned on Buffer Management.* The task for Buffer Management is to get the required main memory and decrease delay and unsatisfiable requirements.

### Chapter 4.7.1 The Structure of Buffer Management

***Principle:***

There has two main buffer management structures:

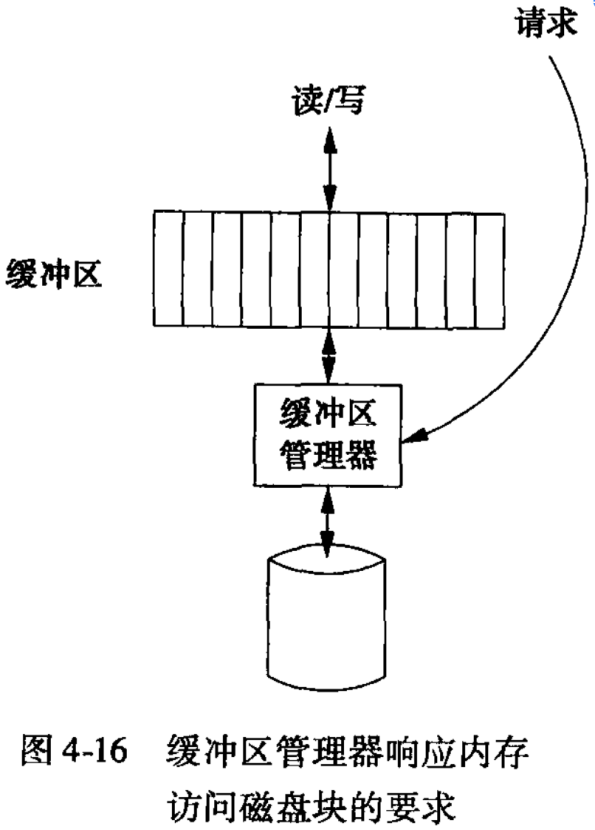
1. In majority Relational DBMS, the buffer management control the main memory directly.
2. Buffer Management is allowed to assign buffers in the virtual memory, it also allowed operation system to put which buffer areas into the main memory and which buffer areas in the ‘Swapping Space’.

***Question:***

No matter for which structure, the Buffer Management should be used to limit the number of buffer areas in order to make them adapt to the memory capacity.

***Solution:***

1. When Buffer Management control main memory directly, but if it exceeds the available space, then we need to return content of buffer into disk to clean buffer areas. If there has no change on the current buffer, then just clean it up; Otherwise if the block has changed, then write it back to the disk.
2. If the number of virtual memory that assigned for the main memory by Buffer Management is much larger than main memory capacity, then when we really use it, then *‘Jolt’* will happens, which means that there have a large bunch of main memory will be move in and move out. Then under this situation, the system will spend a large of time to swap the block other than finish the useful works.



***Conclusion:***

Normally, when initialization DBMS, the number of buffer memory is a parameter setting. In the chapters below, we just assume there has a fixed buffer pool, which is the collection by searching or other Database Operation.

### Chapter 4.7.2 The Strategy of Buffer Management

***Key Point:***

The key point that the Buffer Management do is to choose to throw which block out of Buffer Pool when a new block want to get a buffer area. -> So called *Buffer - Replace Strategy*.

***Least Recently Used (LRU)***

*Rule:*

LRU exchange the blocks that used least recently. This method requires that Buffer Management keeps one table that indicates the last used time of each blocks. It also requires that each time there has one access from database, then there should generate one column.

*Advantage:*

LRU strategy is an effective method, since the least used blocks will get less access than those blocks that have been visited recently.

*Disadvantage:*

LRU needs a large workload to maintain the table information.

***First In First Out (FIFO)***

*Rule:*

In this strategy, when needing a buffer block, then the one which has been occupied for the longest time is cleaned and new block is called and put into the buffer block. In this method, Buffer Management only needs to know when the current block has been filled with. So when read one block from the disk into the buffer block, just generate one column of the table, we should be noted that when the block is being visited, then the column do not need to be modified.

*Advantage:*

Compared with the LRU, FIFO only needs less maintenance.

*Disadvantage:*

FIFO may cause mistake, in the B - Tree index with root block, then it will turn the oldest block in the buffer area. The root block is wrote back to the disk and then read into another buffer block soon.

***Clock Algorithm (The Second Chance)***

*Rule:*

This algorithm is another algorithm of LRU. Clock Algorithm sees the buffer areas as a sequenced circle. The pointer points to one of these buffer areas, if it wants to find a buffer area to put the block exchanged from the disk, then just rotate it based on the clockwise direction.

*Initialization:*

There has a corresponding ‘*flag*’ in each buffer block, it should be 0 or 1. The buffer block with the flag 0 can be easily chosen and wrote the content back to the disk; The buffer block with the flag 1 does not run like this.

Need to pay attention that when the block is read into the buffer area, then its flag will be set as 1. In the same way, when the content of the buffer area has been visited, then its flag will be set as 0.

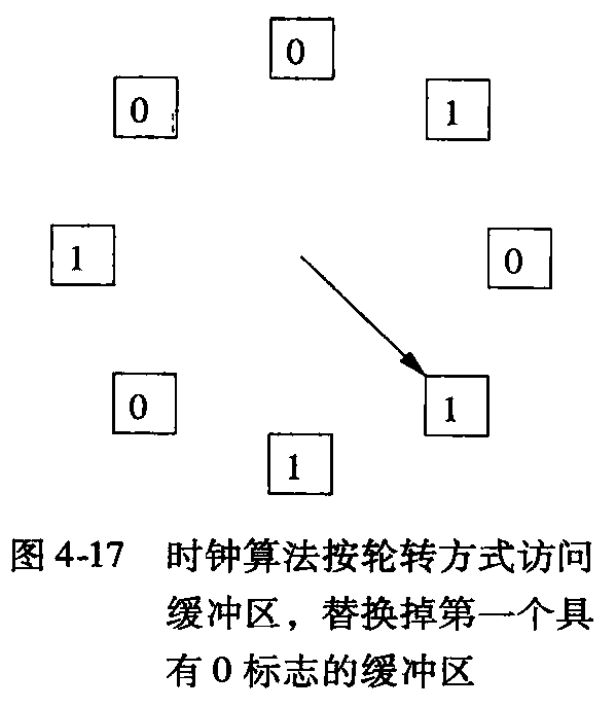
*Procedure:*

When Buffer Management need to read a new block into the main memory and exchange one block out of the main memory, then it rotates according to the clockwise direction, find out the first 1.

When it checks the block whether with flag 1, if satisfies then set it as 0. Therefore, if one buffer block has been visited by the pointer, then its flag has been set as 0. The pointer will continue another completed rotation to find the block with the flag 0. If the block keeps the flag 0, then this block will be thrown out the buffer area.

*Example:*

Analysis from the picture below, when the pointer meets the first two buffer blocks with the flag 1, then it will be set as the flag 0 by the Buffer Management. Then the pointer will continue going through the next buffer block with the flag 0, so the block will be substitutes, and also the flag will be set as 1.



***System Control***

*Rule:*

Query Optimizer or other DBMS partials can be used to remind the Buffer Management to avoid the issue caused by LRU, FIFO or Clock Strategy.

Sometimes if we do not make changes on other blocks that points to the current block, then this block can not be read into the disk. This block is pinned, so any Buffer Management need to make changes on Buffer Substitution Strategy to avoid such situation.

*Example:*

The root of the tree should be defined as pinned block, and force it to keep in the main memory anytime.

For One - Pass Join Algorithm, Query Processor can pin the block with the smaller Relation to ensure that this Relation is in the main memory for the whole time.

### Chapter 4.7.3 The Relationship between Physical Operator Selection and Buffer Management

***Prerequisite:***

Query Optimizer is used to choose the Physics Operators collection to execute the given query. Assume that choosing the Physics Operators can get a number of M buffer blocks for executing each operator.

The problem here is that when we executing query, then Buffer Management does not willing to or can not ensure that it can get M buffer blocks.

***Question:***

1. Does this algorithm can be adapted to the situation of the changes number M of main memory buffers ?
2. How can the Buffer Management using the Buffer Substitution Strategy to influence the extra number of disk I/O ?

***Example:***

***Experiences:***

1. Using the algorithm based on Sort for some operator, then adapt to the change of M is possible. If M decreases, then we can just change the length of sub - table. Of course, when M decreases, then we need to build more sub - tables and during the process of merge, we can not assign each sub - table for one buffer area.
2. If the algorithm is based on Hash, and M decreases, then we can decreases the number of buckets as long as the bucket has not changed too big that can not be put into the assigned memory. But we can not do any reaction to M changes. Once the number of hash buckets is chosen, then it keep the fixed number, if the main memory buffer blocks can not be used any way, then it will be exchanged out.