Buffer Manager

The DBMS performance is due to an efficient buffer management. Buffer is a main memory block pre-allocated to the DBMS, shared among all executing transactions.

The memory is allocated in ages and the size of each one depends on the size of the operating system I/O block. For each buffer page:

- the physical location of the page on memory is found by:
  - file identifier;
  - block number;
- there are state variables with the following funcions:
  - variable *count*, who enumerates the number of transactions that are using the page;
  - variable *dirty bit*, set only if the page has been modified.

In order to access methods to load pages the buffer manager provides the following primitives:

- fix;
- unfix;
- force;
- flush.

Fix Primitive

This primitive have to load the page into the buffer and return a pointer of the page to the transaction who has done the request.

When the primitive ends the page is valid (the transaction can use it) and present in the buffer; *count* state is incremented by 1. If the page is not in the buffer every fix primitive costs a I/O operation.

When a transaction requires a page, first of all, the fix primitive look for that page into the buffer; if it is not there:

- the page can be loaded in a free page of the buffer, if there are any;
- the page can be loaded among pages which are not free but with *count* variable = 0 (pages called victim);
- if the page has *dirty bit*= 1 is written on the disk.

Unfix Primitive

Used to specify that a transaction is no longer using the page, the unfix primitive decrement count variable by 1.

Force Primitive

Provides write operations on the disk in a synchronous mode, for example when a transaction is committed.

Flush Primitive

This primitive transfers pages to disk, indipendently of transaction request and it is done when CPU is in idle time.

Writing Strategies

If Buffer Manager is able to select a page with $count = 0$ as victim he operates in *steal* mode: if transaction committed all runs perfectly, but when transaction rollback there are lots of problems (perform undo operations).

When Buffer Manager is not allowed to select pages belonging to active transactions he operates in *no-steal* mode.

Force and *no-force* strategies are different in the moment when they write active pages on the disk; in the first case pages are written **synchronously** while in the second one pages are written on the secondary memory **asynchronously**. The second approach provides better performances but the most important problem is a failure between the moment in which the transaction ends and the moment in which the buffer write on the disk.

The Buffer Manager performs services provided by the file system, such as:

. create and delete files;

. read/write files;

. open/close files.