8. Buffer Manager

It manages page transfer from disk to main memory and viceversa and it's in charge of managing the DBMS buffer.

DBMS Buffer

It's a large *main memory* block, pre-allocated to the DBMS.
It's sharded among all the executing transactions.
Its memory is organized in pages and their size depend on OS **I/O** block size.

It uses a **Data locality** principle, that is, data referenced recently is likely to be referenced again and uses the empirical approach of **20-80**: 20% of data is read/written by 80% of transactions.

Additionally, the Buffer Manager keeps additional snapshot information on the current content of the buffer.
For each buffer page, it keeps track of its physical location on the disk:

- **file identifier**
- **block number**,
  and its state variables:
- **count:** number of transactions using the page
- **dirty bit:** set if the page has been modified

Buffer Manager primitives

It provides the several primitives to access methods to load pages from disk and viceversa.
It requires shared access permission from the concurrency control manager

Fix primitive

It's used to require access to a disk page.
The page is firstly loaed into the buffer then a pointer to it into the buffer is returned to the requesting transaction.
At the end of the primitive, the page is in the buffer and is valid(that is, it's allocated to an active transaction).
The **count** variable of that page is incremented by 1.
So the way it works is as follows:

1. Fix primitive looks for the page in the buffer

2. If it's present, it return the page to the requesting transaction
3. Otherwise, a page to replace the new one is found by the following criteria:
    - Firstly, among free pages
    - Next, among pages which are not free but with count=0
    - If the page has dirty=1, it's also written on disk
4. Finally it's returned the new page to the requesting transaction

Unfix primitive

It tells the buffer manager that the transaction is no longer using the page.
The **count** variable is also decremented by 1 for that page

Set dirty primitive

It tells the buffer manager that the page has been modified by the running transaction.
It does so by setting the **dirty** variable to 1.

Force primitive

It requires a **synchronous** transfer of the page to disk.
Basically, makes it so the page is written to the disk.
The requesting transaction is suspended until the primitive is done executing, thus it badly affects the performance, specially in RDBMS.
It's usually called before a **commit** to make persistent the changes.

Flush primitive

It transfers pages to disk, independently of transaction requests, for example when pages get replaced when they have the **dirty** bit set to 1.
It runs when the **CPU** is indle.

Buffer Manager writing strategies

Force

All active pages modified by transactions are written in secondary memory during commit.
They are written **synchronously**.
Done so by the **force** primitive

No force

Pages are written on disk **asynchronously** by the buffer manager.
Done so by the **flush** primitive

Steal

The system writes on the disk pages selected to **be replaced** or pages with **dirty bit** set **to 1.** It also **writes** on disk **dirty pages** belonging to *uncomitted* transactions

No Steal

The system is not allowed to select pages beonging to active transactions as victims.

Buffer Manager and File System

The Buffer Manager exploits services provided by the file system, such as open/close and update/delete file syscall