POLITECNICO
DI TORINO

Recap Database Management System (01NVVOV)

Jacopo Nasi
Computer Engineer
Politecnico di Torino

I Period - 2017/2018

5 ottobre 2017

Indice

License

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License.
You are free:

- **to Share**: to copy, distribute and transmit the work
- **to Remix**: to adapt the work

Under the following conditions:

- **Attribution**: you must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work)
- **Noncommercial**: you may not use this work for commercial purposes.
- **Share Alike**: if you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

More information on the Creative Commons website (http://creativecommons.org).

Acknowledgments

Questo breve riepilogo non ha alcuno scopo se non quello di agevolare lo studio di me stesso, se vi fosse di aiuto siete liberi di usarlo.
Le fonti su cui mi sono basato sono quelle relative al corso offerto (**Database Management System (01NVVOV)**) dal Politecnico di Torino durante l'anno accademico 2017/2018.
Non mi assumo nessuna responsabilità in merito ad errori o qualsiasi altra cosa. Fatene buon uso!

1 Database Management System

1.1 Introduction

The DataBase Management System **DBMS** is a software package designed to store and manage databases. The architecture of the system is similar to the one in the figure 1. Since the DB data part can be really big it can't fit

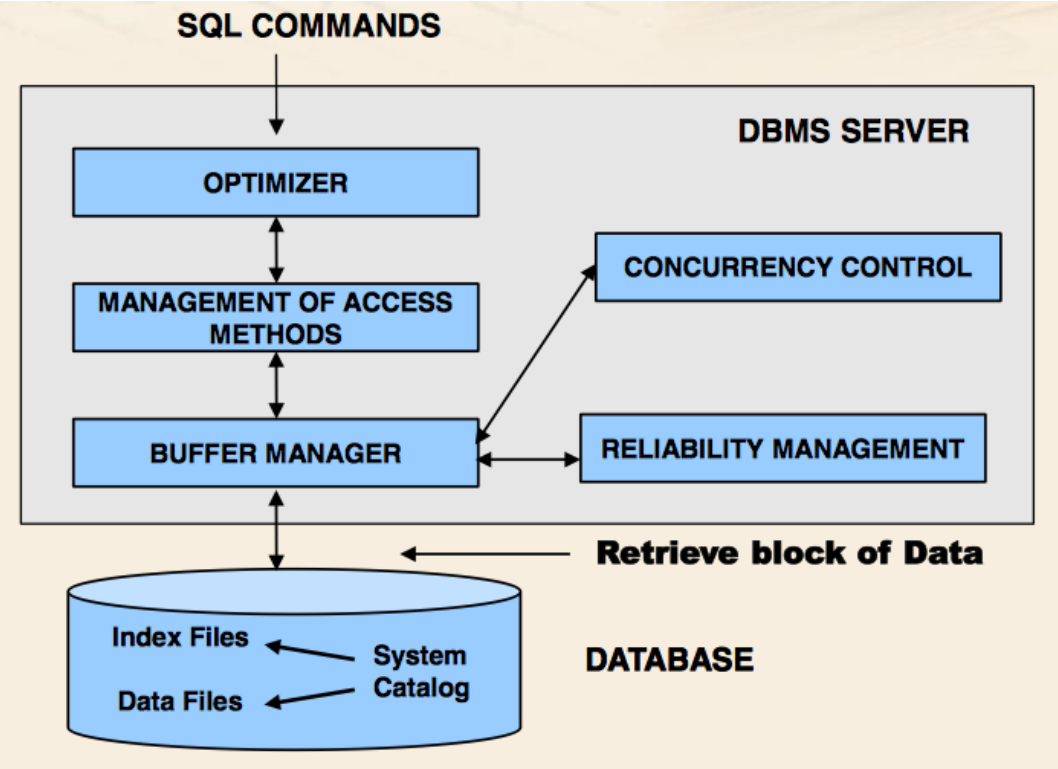

Figura 1: DBMS Architecture

always in the main memory (RAM) and, for this fact, is often stored in the secondary memory, like HDD. For this reason is necessary a system that define the operations to grab and manage the data from the secondary memory.
All the blocks has different behaviours. The **Optimizer** have multiple roles:

- Define an appropriate execution strategy for accessing data to answer queries.
- Receives in input the SQL instructions (DML).
- Check the lexical, syntactical and sematical correctness (not all the errors).
- Translate the query in an internal algebra rappresentation.
- Select the "right" strategy for accesing data.

- Guarantees the **data independence** property in the relation model.

The **Access Method Manager** is used for physical access to data and it implements the strategy selected by the optimizer. The **Buffer Manager** instead manage the page transfert from disk to main memory and vice versa and the main memory portion that is pre-allocated to the DBMS that is shared among many applications. The **Concurrency Control** coordinate the concurrent access to data (important for write operations) to guarantess the consistency of it. The **Realiability Manager** guarantees correctness of the database content duing the system crashes, the atomic execution of a transaction and it exploits auxiliary structures (log files) the correct the database in case of failure.
The **transaction** is an unit of work performed by an application, it's a sequence of one or more SQL RW operation charaterized by *correctness, reliability* and *isolation*. The START of a transaction is typically implicit and coincides with the first SQL instruction. The END instead can be of two differents types, it can be a COMMIT that it means the correct end of a transaction, or with ROLLBACK that it means error during the execution. In this second case the DBMS needs to go back to the state at the beginning of the transaction. The rollback can be of two type suicide, when is required by the transaction, and murder when is required by the system. The transaction have four important properties:

- **A**tomicity
- **C**onsistency
- **I**solation
- **D**urability

Atomicity means that they cannot be divided in smaller units, is not possibile to leave the system in a intermediate state of exec, guarantee by UNDO (undoes all the work perfomed, used for rollback) and REDO (redoes all work performed, used for commit the result in presence of failure). The consistency means that the transaction execution should not violate integrity constraints on a database, in case of it the system will perform solution to correct the violation. The system can be considered Isolated when the execution of a transaction is indipendent of the concurrent execution of other transaction, everything is enforced by the Concurrency Control block. The last properties means that, in presence of failures, the effect of a commited transaction IS NOT LOST, it guarantees the reliability of the DBMS and is enforced by the Reliability Manager block.

1.2 Buffer Manager

This block have a real important behaviour, it manages page transfer from disk to main memory and it's in charge of managing the DBMS buffer. The

operation of the pages trasnfert is the bottleneck of every system and this is why this block is really important. increasing the performance of this operation could really improve the speed of the entire system.
The buffer is:

- A large main memory block.
- Pre-allocated to the DBMS.
- Shared among executing transactions.

this part is organized in pages where the size depends on the size of the OS I/O block. There are two empirical law often used for hte management strategies:

1. Data Locality: Data referenced recntly is likely to be referenced again.
2. 20-80: The 20% of data is RW by 80% of transaction.

The buffer manager keeps additional snapshot information on the current content of the buffer, it shot, for every page, the physical location of the page on the secondary memory (file identifier and block number) and two state variables, one that count of the number of trasn using the page in that time (count), and the dirty bit that is set if the page has been modified.
It provides different access methods to load pages from disk and vice versa:

Fix Primitive used by transactions to require access to a disk page, after the page is loaded into the buffer a pointer is returned to the requesting transaction and the Count is incremented by 1. This procedure requires an I/O operation only id the page is not already in the buffer. There are two behaviour:

- Page already in buffer: Return the pointer to the data.
- Page not in buffer: It search a place for the page.
    1. Free pages
    2. Not free pages, Count=0; if the data is dirty it performs a synchronous write on the disk.

Unfix Primitive it tells the buffer manager that the transaction is no longer using the page and it decrease the Count.

Set Dirty Primitive is tells the buffer manager that the page has been modified by the running transaction and it sets the dirty bit to 1.

Force Primitive it requires a synchronous transfert of the page to the disk, when this operation is performed the transaction is suspended.

Flush Primitive is an autonomous transfert of the pages on the disks, is internal to the buffer manager and is runned when the CPU is not too much loaded. It transfer the page that are not valid (count=0) or not accessed since long time.
The are four writing strategies:

- **Steal**: The BM is allowed to select a locked page with Count=0 as victim. It writes on disk the dirty pages belonging to uncommitted trans. It can be undone.
- **No Steal**: The BM is not allow to steal.
- **Force**: All the pages are synchronous written on the disk during the commit operation.
- **No Force**: The pages are written asynchronously with the Flush Primitive.

The mostly used solution is **steal/no force** because of its efficency. The no force provides better I/O performance, steal may be mandatory for queries accessing a very large number of pages.

File System the BM is using services provided by the file system:

- Create/Delete of a file.
- Open/Close file.
- Read: It provides a direct access to a block in a file and it requires File Identifier, Block number and buffer page where to save data.
- Sequential Read: It provides seq. access to a fixed number of blocks in a file, it requires file identifier, strating block, number of blocks to be readed and the starting page for saving.
- Write and Sequential Write.
- Directory management.