Summary of Architecture of a Database System

By S. Xiao Fernández Marín.

1 Summary

When a query enters to the *Client Communications Manager* (CCM), responsible for communicating with the clients and remembering the connection state for the caller, using the remote and local client protocols, where the unique sockets, name, pipes or connecting to network TCP or other protocol are done. The CCM sends it to the *Relational Query Processor*, where the query is parsed. It sends it to *Admission Control* whether it decides if the query can be executed. After the query is admitted, it can be understood and it checks with the *Catalogue Manager*, the part where is checked that it has permissions to use the tables it needs. Finally, it rewrites the query and converts it into an internal query plan. this is sent to the plan executor (how you execute the query).

There are also different options for DBMS process models. **OS (Operative System) Process:** Where each process has at least 1 thread and cannot be overwritten as each one has its own address space. **OS Thread:** A OS program without additional private address space. The access to the common memory so is faster, but there is a risk of overwriting. **Lightweight Thread Package:** The use of a library, independent of the OS, so does not give it support. **DBMS Client:** Software component that implements the API used by the app to communicate with the DBMS. And **DBMS worker:** The ones that execute the queries, there are three processes model of workers: OS process per worker: DBMS worker mapped as an OS Process. OS thread per worker: A single multi-threaded process hosts all the DBMS workers. Process Pool: Variant of the *OS process per worker*. It pre-allocates the threads and reuses them.

The independence of the whole DBMS Worker is not possible because the queries request must access the following common resources. **Catalogue:** Table of definitions, used in DB, passwords, users, etc. **Lock Table:** Implement DB locking semantics. And the **Disk I/O Buffers**, in where we find: Log: All the changes we have done, an array of entries. DB I/O request: Memory where we buffer the things we read or write.

All modern systems are based on parallel architecture, and there are different isolation levels. **Shared memory:** All processors can access the same RAM and disk. **Shared disk:** Some cores share disk. **Shared nothing:** Systems connected by a network.

Now we will talk about addressing each of the main DBMS components, starting with how a query is processed, and differentiating the *Data Definition Language (DDL)*, where we define tables and does not need optimization, from *Data Manipulation Language (DML)* where we handle tables. It consists of 4 stages: **Parsing and authorization:** Checks the query is correct, identify the data and converts SQL to an internal structure. After that, it checks that the user is allowed to execute it. **Rewiring/Transformation:** Rewrite the query for it to be simpler. **Optimization:** Find the best way of resolving the query. **Execution:** Compiled to machine or VM code.

In this last part, it enters the storage, so it is managed in two different ways. **Direct/raw disk I/O**, that has full control of disk usage or **Filesystem** that has less control of the OS buffering.

To manage the buffer, the shared buffer pool of each DBMS in which the data is stored before writing it in the disk is dynamically allocated and organized as an array of frames.

The transactions are the way of accessing the storage. They have to be ACID (Atomicity, Consistency, Isolation, and Durability) and has 4 main different components: Lock manager for concurrency control, log manager for recovery, buffer pool for storing I/Os DB and access methods for organizing the data on disks.

The transactions have to be **serializable**, a sequence of interleaved transactions execute as if they were executed one by one, have **concurrency control** using different techniques such as 2PL, MVCC or OCC. And make decisions between **locking or latching**, different locking mechanisms,

where one of the main differences is that locks are kept in a table and located using hash tables and latches are stored in memory near the resources protected.

Different isolation levels exist in transactions, to increase concurrency by providing declarative definitions of the consistency and implementation like **read uncommitted**, **read committed**, **repeatable read** or **serializable** between others. The default of isolation depends on the DBMS.

There are also **log managers**, who is in charge of maintaining durability and recovery for making it possible to abort transactions. There also exist indexes, that are physical storage structures for accessing data in the database and are managed by transnational schemes (the most typical one: *B+tree*) by using locking and logging and handling it with latching instead of 2PL, as it is over-killed.

On the storage of a DBMS, we can find the catalogue, where is stored the metadata of the database (tables, name of tables, statics, etc) and it is accessed by all the queries. The memory allocation can be handled by allocating just one time in the front or using pointers instead of copying data.

The replication of a DB can be **physical** duplicating the whole DB, **trigger-based** by placing triggers on the DB and record modifications or **log-based**, by intercepting logs writes and delivering it to the remote system.

Finally, the monitoring and administration tools of a DB maintenance are **Automatic statistics gathering**, **Physical Reorganization and Index Construction**, **Backup/Export** for a dump the DB to backup, **Bulk Load** faster than INSERT statements or **Monitoring, Tuning, and Resource Governers** provide if a query consume more resources than the rest of them.

2 Questions not answered by this text

The user can also have a process pool. It can open a connection with the server and if it is closed, it does not close the thread and then it reuses it.

There are also some systems where they differentiate between parsing (using a parse tree or an abstract syntax tree) and resolving (semantic verification of the query) or where they don't differentiate between resolving and rewriting, rewriting and optimization or optimization and execution.

As the information in the catalogue is not directly accessible to the user, you can find it by using the command *SELECT ... FROM INFORMATION_SCHEMA*.

3 What has changed since this was written

Since 2007, the OS thread has added more protection, as now, for example, each time you open a tab in Chrome, it opens a new process because if one tab crashes, the other ones can still run. The same thing happens with the DB and the connection.

It has also changed the fact that when this paper was written, there was no parallel execution. Not as today, that all the processes have several cores.

The disks are now just one single disk with more complex performance and the SSDs are replacing the spinning disks so we remove the problem of random access.