Architecture of a Database System Joseph M. Hellerstein, Michael Stonebraker and James Hamilton

A short summary by Platon Karageorgis

The birth of a query constitutes the initialization of a series of events that end with the termination of the query. The user who executes the query expects from the system ideally the resulting tuples. To him, due to the modern technological level of the Databases Management Systems or DBMS, the process is probably a very abstract function that does not concern him. The journey of the query though, can guide the person that investigates it through the whole system and it is something the paper considers. The moment the query is executed by the client, the Client Communications Manager establishes a connection and notifies the process manager. The process manager's prime job is to make sure that every thread, either it is an operating system thread or a lightweight one, finishes smoothly without interrupting the system. It is responsible for dealing with concurrency issues by using techniques such as the memory efficient process pool or facing thrashing difficulties by enforcing admission policies. In addition, since the idea of parallel architecture was introduced, the above problems became much harder and caused the need of new implementations. The shared nothing, the shared disk and the shared memory system were the result of parallelization's rise and they represent different implementations that are all very common among the database systems. To continue, after going through all these protocols, the process manager hopefully will confirm that everything ran smoothly and will set a time for the query to begin. When it does, the relational query processor first parses the query and then is in charge of running a series of protocols in order to improve it such as query rewriting, query optimization and finally query execution. The query rewriter's job is to simplify the query without altering anything that would return a different result, whereas the query optimizer is responsible for shifting an interior query representation, into a well-ordered and methodical plan for executing the query. As soon as the query plan is ready, it is presented to the query executor who initiates iterations based on the old-fashioned object oriented iterator. Finally, the iterator runs through B+- trees -which is the supported data structure of every system- and its completion signs the end of the query processing. Moving on, the query needs a permission to access the data that is granted by the Transactional Storage Manager. Once that happens, the tuples are starting to be computed and the storage manager is in charge of organizing them in respect of protocols such as spatial control that holds significant information for the virtual and the physical location of the data. The storing of the data indicates the termination of the query. The paper investigates thoroughly every step that was mentioned, since its mere goal is to discuss the architectural side of

a DBMS. Simultaneously it adds a lot more information. It takes into account its historic contribution into computer science. It looks ahead for the reader every time challenging terms are needed by ensuring that they are clarified beforehand and it gives multiple examples when it comes to the actual usage of the techniques discussed in the real world. Furthermore, it does not forget to include the time variable, since in the modern era everything is changing daily, by pinpointing when a method is becoming obsolete and underlines when a subject does not have much ground for evolution. All these result to a compact analysis of the design of a DBMS and everything that revolves around it.

Which questions aren't answered by this text?

In my opinion the paper is very solid. It sets a very clear goal which is to cover the topic of the architecture of database systems and it does that in a way that even a person that is not very related can understand by including definitions when necessary. It has a concrete and organized plan throughout every chapter. Finishing the reading of this paper I feel covered and without any kind of question.

What has changed since this was written?

The paper was written in 2007 which is 15 years ago. Obviously a lot of things have changed. I think the concurrency methods and parallel programming in general should be much more evolved, since its one of the hottest topics of programming in the 21st century. Also the storing methods should be different. The paper mentions flash disks as a new thing that might turn out to be very popular in the future and it is definitely on point because they are widely used.