Reading Assignment 1 – *Anatomy of a Database System*

José Filipe Barbosa de Carvalho (josé.carvalho@fe.up.pt)

26th October 2007

Advanced Database Systems
Technische Universität Wien,
Karlsplatz 13, A-1040 Wien AUSTRIA

Abstract: This text wants to resume the fundamental ideas in [1], after a careful reading. It also presents things that author didn't well understand and his personnel opinion about it.

1 Introduction

Nowadays Database Management Systems (DBMSs) are complex and an important component in almost information systems. However it is difficult find good information about database systems architectures and about specific internal implementations, because the community of people involved in development of database systems is relatively small and because much information is a commercial secret.

The paper [1] presents the challenges to design and to implement a simple database system, presenting the main components that compound the database application. To maintain text simple to understand, many of new features and extensions, like stored procedures, object-oriented databases or XML, are omitted. The paper uses a historical perspective, showing not only how databases evolved in the last 30 years, but also related aspects, like evolution of hardware and operation systems (OS).

2 Important ideas and results of the paper

The paper discusses, in a simple and concise way, the architectural and implementing issues of database systems. First it shows overall architecture of DBMS processes and how they are implemented in different hardware and operating systems.

After that, it presents the main components that a DBMS must have (see figure 1), detailing a lot of issues in their development. The four main components, and respective subcomponents', of a database management system are:

- Process Manager (Admission Control and Connection Manager);
- Query Processor (Parser, Query Rewrite, Optimizer and Executor);
- Transactional Storage Manager (Access Methods, Buffer Manager, Lock Manager and Log Manager);
- Shared Utilities (Memory Manager, Disk Space Manager, Replication Services and Admin Utilites).

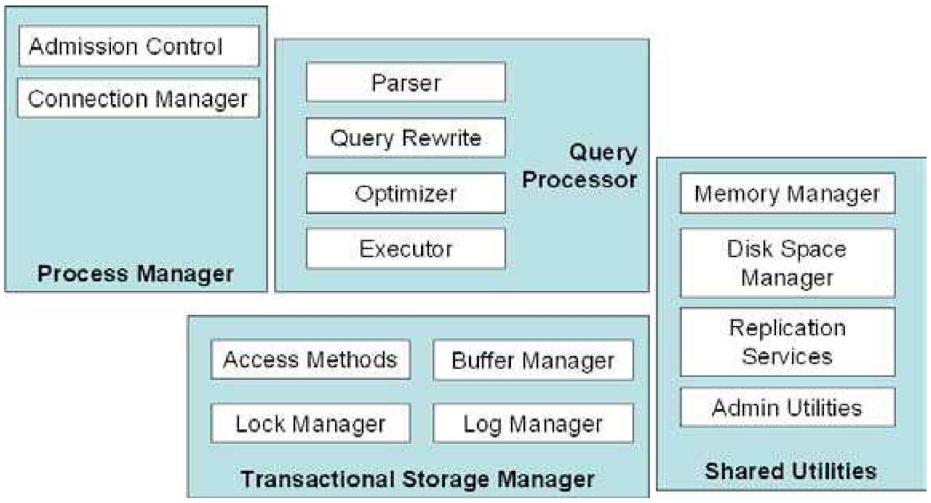

Figure 1 – Main components of a Database Management System

In the first part of text, authors introduce the subject, contextualize and present structure of paper. After that, begin second chapter, which discusses process models and hardware architectures. The papers show possible decisions when we build a multi-user server, based in hardware and operating system used. So, it discusses the use of OS processes or OS threads, and the advantages/disadvantages of using specific DBMS threads. Assuming uniprocessor hardware and high-performance OS threads, the text presents three process models:

- One process per client connection;
- One server process, with multiple threads;
- One server process, with multiple threads, and I/O processes to provide asynchronous I/O features.

In this part, [1] also propose how to passing data across threads. After that it shows how to map between DBMS threads and OS processes and the current implementations of this in some commercial databases.

In second chapter are also presented the challenges of parallelism and of memory coordination in process models, when platforms have more than one processor. It discusses four different architectures:

- Shared Memory architectures: all processors can access the main memory and disks with the same performance;
- Shared Nothing architectures: a set of single-processor machines that communicate over a high-speed network interconnect;
- Shared Disk architectures: all processors can access the same disks with about same performance, but can’t access each other’s main memory;
- Non-Uniform Memory Access architectures: existence of a shared memory where the time required to access some remote memory is much higher than the time required to access local memory.

All the four architectures have their trade-offs, that database administrator (DBA) must understand to choose the appropriate architecture for a given scenario. Nowadays marketplace supports a mix of Shared-Nothing, Shared Memory and Shared-Disks architectures.

The chapter two of [1] presents also the admission control component, which can be have two layers: the first one ensures that number of clients is kept below a threshold; the second layer is execution admission controller, that must be implement within the core DBMS query processor, to decide whether a query is postponed or begins immediately, depending on resources that it will use. To finalize, second chapter shows the standard practice in modern DBMSs to accomplish issues described in chapter.

The chapter three begins with basic considerations about storage models. It discusses spatial control of data in disks and also temporal control, that is, when write and read blocks to and from disk. One important subject is if database should use OS services or use low-level operations, with direct access to the disk. The second choice complicates the implementation but can improve performance, because database application knows what is better for their data (like store all *tuples* of a table together) and OS not. Today, because commercial file systems have evolved is common practice allocate a large file in file system and DBMS control the placement of data within this file using special OS interfaces. However direct access to disks remains a common high-performance option in most of database systems. This chapter also presents buffer management issues, talking about the page replacement policies like LRU and Clock.

The fourth chapter is all about query processor. It presents their subcomponents: parser, query rewrite, optimizer and executor. Query processor is responsible “to take a declarative SQL statement, validates it, optimizes it into a procedural dataflow implementation plan, and (subject to admission control) executes that dataflow” [1], which will generate a result that will be fetched by client program.

The query processor parser is the subcomponent that validates SQL statement, converting it into an internal format and checks if user has permission do this operation. It makes use of catalog manager to do these tests. And what is catalog manager? Is a kind of metadata that stores information about data in system, like table, users, indexes, and so on.

The query rewrite receive as input a query and optimizes it, only by rewriting of that expression. It doesn’t change the internal format of query. For example, it does view expanding or simplifies some logical predicates. The function of optimizer is to create an efficient query plan (it is like a diagram flow that starts from base relations and after a set of operations has the same meaning that original SQL statement) to execute it. To create this “efficient plan” exist several techniques to minimize the number of operations and data used to perform the queries. The executor is like a runtime interpreter: it follows the query plan did by optimizer, recursively invoking operators existing in the plan. The paper also discusses the iterator model employed by many modern executors, and many challenges of their implementation, like if it is better to store iterators and their data in buffer pool or in memory heap.

In final part of chapter four it also refers access methods, one subcomponent of Transactional Storage Manager. The access methods are the routines for manage access to diverse disk-based data structures, which usually include unordered files of tuples, and various kinds of indexes. Indexation is a technique to optimize the speed in the access of large amounts of data. In text [1] said that all commercial database systems including B+-tree indexes and heap files.

The fifth chapter talks about transactions and two topics associated: concurrency control and recovery. There is explained why databases appear as enormous and monolithic pieces of software: the subcomponents of transactional storage manager are really intertwined:

- A lock manager for concurrency control;
- A log manager for recovery;
- A buffer pool to perform database I/Os;
- Access methods to organize data on disk.

This text refers what it is an ACID transaction, a mnemonic for Atomicity, Consistency, Isolation and Durability. There are four characteristics that all transactions must have, however they are not formally defined, but only a general reference. It also noted the four isolation levels defined ANSI SQL standard: Read Uncommitted, Read Committed, Repeatable Read and Serializable. The first three don’t guarantee serialization of transactions, but provides more concurrency.

The lock manager is responsible to manage concurrency, namely to ensure that “a sequence of interleaved actions for multiple committing transactions correspond to some serial execution of the transactions” [1], using, for example, the two-phase locking (2PL). Database locks correspond to some resource or data that we want ensure exclusive access for some time. For example, we must lock one table if we want delete some of its tuples, to ensure that another transaction don’t change our goals for our transaction. Normally databases also support latches, another kind of exclusion mechanism, but to provide access to internal DBMSs data structures.

The log manager ensures the durability of transactions and provides methods to rollback in aborted transactions, to maintain atomicity property. It registers a sequence of log records on disk and uses several data structures in memory to provide this functionality. Many DBMSs make uses of ARIES protocol to implement this subcomponent.

To finalize fifth chapter also discusses the challenges of locking and logging indexes and the interdependencies of transactional storage.

3 Things not well understood

I haven't troubles in understood almost of this paper. I have a strong preparation in Operating Systems and Database topics, which I got in my earlier courses. Namely, it is important have specific knowledge about databases, as what is subqueries, indexes, views, schema constraints, and so on, to understand this paper. I also have good background in algorithmic and data structures, like B++ tree or hash tables.

Of course I didn't know some jargon like "Thrash" (appeared in 14th page) or some technologies like IBM SP2 (appeared in 12th page). But with a simple search in Google or Wikipedia I understood, in a general way, these simple concepts.

The page replacement policies (like LRU and Clock) are mentioned in second lesson of Advanced Database Systems course, otherwise maybe I couldn't understand section 3.3.

When paper discuss Iterator model of Executor component (section 4.4), I couldn't understand how exchange iterators provided support for parallel query: maybe I should read the additional bibliography mentioned in the text – but the text don't have list of references.

I also didn't understand well the differences between the three approaches in latching B+-Trees (section 5.4.1): conservative, latch-coupling and right-link schemes. Maybe if I read more about B+-Trees and Indexes and read the references showed in that section.

4 Things that I like and I didn't like in the paper

The paper is written in friendly way, explaining the concepts in a historical perspective and with a lot of specific examples, revealing some implementation details of the most popular databases, as Oracle, Microsoft SQL Server and DB2. I like too much the paper because it details the internal architecture of databases, a totally new subject for me. My previous experience in databases systems is more focused in using them, as application programmer: designing databases schemas, implementing SQL queries and stored procedures, tuning performance of database, and so on. This paper gives me another image of databases systems, showing the major challenges in their implementations.

Sometimes the paper refers the implementation used in some famous databases with some degree of uncertainly (see, for example, the commentary about using of top-down optimizers in Microsoft SQL Server, page 25). Is questionable do this in a scientific paper, but I think that is justifiable: implementation of commercial database systems is a valuable secret.

But the paper is written in 1998 and we are at 2007. Almost ten years has passed, and informatics topics evolving too fast. Probably the paper has some out of date expressions. For example, I think that is strange that Oracle (one of most used databases) installation for UNIX maintains uses, by default, the one process per connection architecture (see page 5 of [1]): maybe that sentence is out of date or maybe another reason explains this (like as explained in page 10, where explains that one process in Linux have only available 3GB of memory).

The document is well-structured and is a good resume of implementation issues of database systems. But in some parts I think that the paper has some needless low-level details: for instance, the description of locks and latches APIs (in section 5.2) is more or less boring. If authors used some abstraction this part is more interesting and less messy.

Another critic is that version of paper is incomplete, the page forty two ends with an incomplete sentence and I can't see the references noted in the text, and I couldn't find the complete version at Internet.

5 Conclusion

This paper is a good point to start understanding the internals of database management systems, as many of challenges to design and to implement it. Of course some topics are very general, but paper presents a good overview to the subject, that we can explore and go deeply with further readings.

6 Bibliography

[1] HELLERSTEIN, Joseph M.; STONEBRAKER, Michael; "Anatomy of a Database System." (Section 1-4) *Readings in Database Systems*, 1998