6.830/6.814 — Notes* for Lecture 4: Database Internals Overview

Carlo A. Curino

September 22, 2010

1 Announcements

- Problem Set 1 is due today! (For next time please submit some open non-esoteric format .txt .rtf .pdf)
- Lab 1 is out today... start it right away, is due in 8 days! Do not copy... we perform automatic checking of plagiarism... it is not a good gamble!
- Projects ideas and rules are posted online.

2 Readings

For this class the suggested readings are:

- Joseph Hellerstein, Michael Stonebraker and James Hamilton. Architecture of a Database System. Online at: `http://db.cs.berkeley.edu/papers/fntdb07-architecture.pdf`

It is a rather long paper (don't be too scared by the 119 pages, the page format makes it look much longer than it is) that is in general worth reading, however we only require you too read sections: 1, 2 (skim through it), 3, 4 (up to subsection 4.5 included), 5. You can also skim through section 6 that we will discuss later on. Probably doesn't all make sense right now – you should look at this paper again to this paper through the semester for context.

3 A bit of history

Complementing Mike's historical overview...Projects ideas and rules are posted online.

*These notes are only meant to be a guide of the topics we touch in class. Future notes are likely to be more terse and schematic, and you are required to read/study the papers and book chapters we mention in class, do homeworks and Labs, etc.. etc..

1970's : Several camps of proponents argue about merits of these competing systems while the theory of databases leads to mainstream research projects. Two main prototypes for relational systems were developed during 1974-77.

- Ingres: Developed at UCB by (including guess who? Stonebraker and Wong). This ultimately led to Ingres Corp., Sybase, MS SQL Server, Britton-Lee, Wang's PACE. This system used QUEL as query language.
- System R: Developed at IBM San Jose (now Almaden) and led to IBM's SQL/DS & DB2, Oracle, HP's Allbase, Tandem's Non-Stop SQL. This system used SEQUEL as query language (later SQL). Lots of Berkeley folks on the System R team, including Gray (1st CS PhD @ Berkeley), Bruce Lindsay, Irv Traiger, Paul McJones, Mike Blasgen, Mario Schkolnick, Bob Selinger , Bob Yost.

Early 80's : commercialization of relational systems

- Ellison's Oracle beats IBM to market by reading white papers.
- IBM releases multiple RDBMSs, settles down to DB2. Gray (System R), Jerry Held (Ingres) and others join Tandem (Non-Stop SQL), Kapali Eswaran starts EsVal, which begets HP Allbase and Cullinet
- Relational Technology Inc (Ingres Corp), Britton-Lee/Sybase, Wang PACE grow out of Ingres group
- CA releases CA-Universe, a commercialization of Ingres
- Informix started by Cal alum Roger Sippl (no pedigree to research).
- Teradata started by some Cal Tech alums, based on proprietary networking technology (no pedigree to software research)

Mid 80's :

- SQL becomes "intergalactic standard".
- DB2 becomes IBM's flagship product.

1990's:

- Postgres project at UC Berkeley turned into successful open source project by a large community, mostly driven by a group in russia
- Illustra (from Postgres) → Informix → IBM
- MySQL

2000's:

- Postgres → Netezza, Vertica, Greenplum, EnterpriseDB...
- MySQL → Infobright
- Ingres → DATAllegro

System R is generally considered the more influential of the two – you can see how many of the things they proposed are still in a database system today. However, Ingres probably had more "impact" by virtue of training a bunch of grad students who went on to fund companies + build products (e.g., BerkeleyDB, Postgres, etc.)

4 Introduction

Figure 1 shows the general architecture of a database.

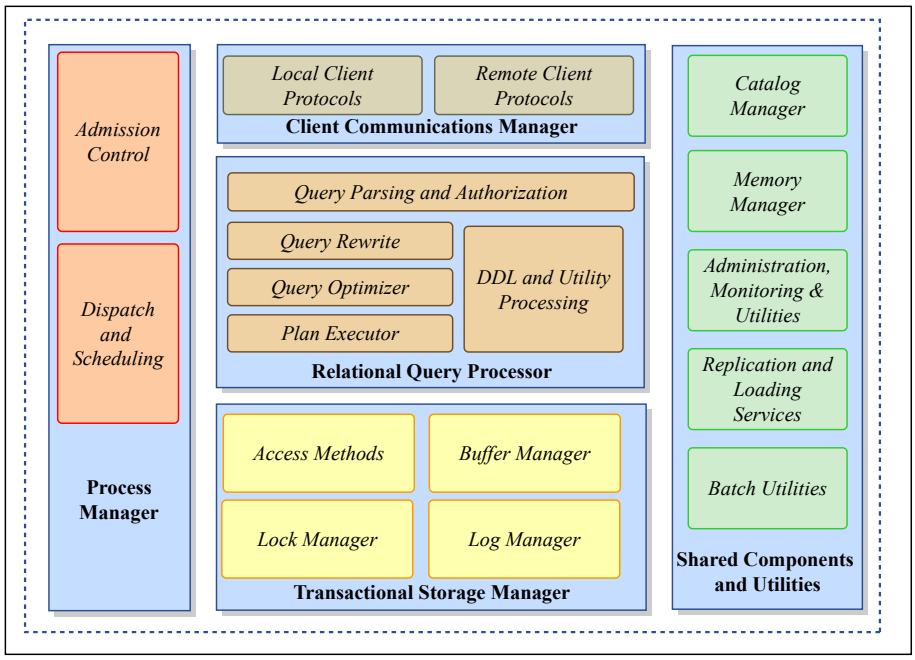

Image by MIT OpenCourseWare.

Figure 1: Architecture of a DBMS

Today we will mainly look at the big picture, and go through the relational query rewriting and execution, the following lessons will focus on each of the pieces in more details.

Show flow of a query

5 Process Models

Parallelism is a key to performance, in particular when I/O waits might stall computation. To maximize throughput you need to have enough stuff going on in parallel to avoid waiting/stalling.

Process models:

- Back in the days there was no good OS thread support, DB pioneered this ground (also due to the need of supporting many OSs)
- Process per DBMS worker (need for shared memory [**ASK**: is it clear why we need to share across multiple workers?], context switch is expensive, easy to port, limited scalability)
- Thread per DBMS worker (great if good OS thread support, or using DBMS separate implementation of threads... pro: portability, cons: duplicate functionalities)
- Process/Thread pool, and scheduling/allocation of DBMS workers to processes or threads.

6 Parallel Architecture

- Shared Memory: typically inside one machine, for large installation high costs. All process models are applicable. Great for OLTP, many implementation form almost every vendor.
- Shared Nothing: typically as a cluster of nodes. Require good partitioning, which is easier for OLAP workloads (Teradata, Greenplum, DB2 Parallel Edition,...).
- Shared Disk: cluster of nodes with a SAN. Simple model, because every node can access all the data, but requires cache-coherence protocols. Oracle RAC, DB2 SYSPLEX.
- NUMA: not that common, we will not discuss. (Often DBMS treat it as either shared nothing or shared memory, depending how non-uniform it is).

Different failure modes... Partial failure is good to have when possible.
We will go back to parallel architectures later on, and dis

7 Query Processing

Query parsing (correctness check)
Query admission control / authorization

7.1 Query Rewrite:

View Rewrite

Remember the other day schema:

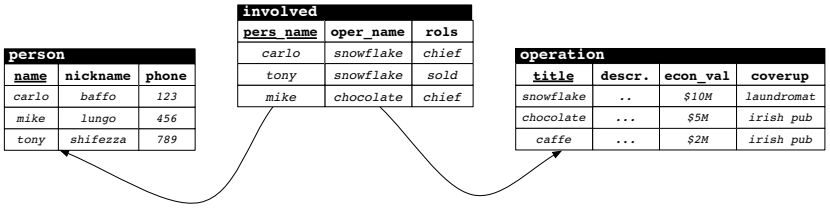

Figure 2: Simple Schema for a portion of our Mafia database.

What are views? A "named-query", or a "virtual-table" (sometimes materialized).

```
CREATE VIEW nick-cover AS
SELECT  nickname, coverup_name
FROM operation o, involved i, person p
WHERE p.name = i.person AND
      i.oper_name = o.name AND
      o.econ_val <= 5M;
```

nick-cover

nickname	coverup
baffo	laundromat
lungo	irish pub
schifezza	laundromat

Figure 3: Simple External Schema for a portion of our Mafia database.

```
SELECT  nickname
FROM nick-cover nc
WHERE nc.coverup_name="laundromat";
```

After view rewriting:

```
SELECT  nickname
FROM (
 SELECT  nickname, coverup_name
 FROM operation o, involved i, person p
 WHERE p.name = i.person AND
       i.oper_name = o.name AND
```

```
      o.econ_val <= 5M
  ) as n
WHERE n.coverup_name="laundromat"
```

7.1.1 Contraint Elimination / Logical Predicates manipulation

Another important step of query rewriting consists of *Constant Elimination* and *Logical Predicates manipulation*

```
WHERE (a > 50+57 OR a =107) AND b > 105 and a=b AND b < 108
```

becomes (after constant elimination, and logical predicate manipulations):

```
WHERE a = 107 and a = b and b = 107
```

7.1.2 Subquery Flattening

As Mike mentioned the last class another key step is *Subquery flattening* (Not every optimizer will successfully do this, so you should always try to think of a non nested query if you can find one):

```
SELECT  nickname
FROM operation o, involved i, person p
WHERE p.name = i.person AND
    i.oper_name = o.name AND
   o.econ_val <= 5M AND
   o.coverup_name="laundromat";
```

7.1.3 Semantic Optimization (Integrity Constraints)

Sometimes, knowledge of integrity constraints, in particular, foreign keys, can be leveraged to avoid performing joins. As an example consider the following query:

```
SELECT  nickname
FROM operation o, involved i, person p
WHERE p.name = i.person AND
    i.oper_name = o.name AND
```

If we know from the foreign keys that every person that appear in the involved tuple is involved in some operation, we can skip the join with operations.

6.830 / 6.814 Database Systems

Fall 2010