Chapter 2: Traditional RDBMS Systems

Introduced by Michael Stonebraker

Selected Readings:

Morton M. Astrahan, Mike W. Blasgen, Donald D. Chamberlin, Kapali P. Eswaran, Jim Gray, Patricia P. Griffiths, W. Frank King III, Raymond A. Lorie, Paul R. McJones, James W. Mehl, Gianfranco R. Putzolu, Irving L. Traiger, Bradford W. Wade, Vera Watson. System R: Relational Approach to Database Management. *ACM Transactions on Database Systems*, 1(2), 1976, 97-137.

Michael Stonebraker and Lawrence A. Rowe. The design of POSTGRES. *SIGMOD*, 1986.

David J. DeWitt, Shahram Ghandeharizadeh, Donovan Schneider, Allan Bricker, Hui-I Hsiao, Rick Rasmussen. The Gamma Database Machine Project. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), 1990, 44-62.

In this section are papers on (arguably) the three most important real DBMS systems. We will discuss them chronologically in this introduction.

The System R project started under the direction of Frank King at IBM Research probably around 1972. By then Ted Codd's pioneering paper was 18 months old, and it was obvious to a lot of people that one should build a prototype to test out his ideas. Unfortunately, Ted was not a permitted to lead this effort, and he went off to consider natural language interfaces to DBMSs. System R quickly decided to implement SQL, which morphed from a clean block structured language in 1972 [2] to a much more complex structure described in the paper here [1]. See [4] for a commentary on the design of the SQL language, written a decade later.

System R was structured into two groups, the "lower half" and the "upper half". They were not totally synchronized, as the lower half implemented links, which were not supported by the upper half. In defense of the decision by the lower half team, it was clear they were competing against IMS, which had this sort of construct, so it was natural to include it. The upper half simply didn't get the optimizer to work for this construct.

The transaction manager is probably the biggest legacy of the project, and it is clearly the work of the late Jim Gray. Much of his design endures to this day in commercial systems. Second place goes to the System R optimizer. The dynamic programming cost-based approach is still the gold standard for optimizer technology.

My biggest complaint about System R is that the team never stopped to clean up SQL. Hence, when the "upper half" was simply glued onto VSAM to form DB2, the language level was left intact. All the annoying features of the language have endured to this day. SQL will be the COBOL of 2020, a language we are stuck with that everybody will complain about.

My second biggest complaint is that System R used a subroutine call interface (now ODBC) to couple a client application to the DBMS. I consider ODBC among the worst interfaces on the planet. To issue a single query, one has to open a data base, open a cursor, bind it to a query and then issue individual fetches for data records. It takes a page of fairly inscrutable code just to run one query. Both Ingres [11] and Chris Date [3] had much cleaner language embeddings. Moreover, Pascal-R [9] and Rigel [8] were also elegant ways to include DBMS functionality in a programming language. Only recently with the advent of Linq [7] and Ruby on Rails [5] are we seeing a resurgence of cleaner language-specific enbeddings.

After System R, Jim Gray went off to Tandem to work on Non-stop SQL and Kapali Eswaren did a relational startup. Most of the remainder of the team remained at IBM and moved on to work on various other projects, include R*.

The second paper concerns Postgres. This project started in 1984 when it was obvious that continuing to prototype using the academic Ingres code base made no sense. A recounting of the history of Postgres appears in [10], and the reader is directed there for a full blow-by-blow recap of the ups and downs in the development process.

However, in my opinion the important legacy of Postgres is its abstract data type (ADT) system. User-defined types and functions have been added to most mainstream relational DBMSs, using the Postgres model. Hence, that design feature endures to this day. The project also experimented with time-travel, but it did not work very well. I think no-overwrite storage will have its day in the sun as faster storage technology alters the economics of data management.

It should also be noted that much of the importance of Postgres should be accredited to the availability of a robust and performant open-source code line. This is an example of the open-source community model of development and

maintenance at its best. A pickup team of volunteers took the Berkeley code line in the mid 1990's and has been shepherding its development ever since. Both Postgres and 4BSD Unix [6] were instrumental in making open source code the preferred mechanism for code development.

The Postgres project continued at Berkeley until 1992, when the commercial company Illustra was formed to support a commercial code line. See [10] for a description of the ups and downs experienced by Illustra in the marketplace.

Besides the ADT system and open source distribution model, a key legacy of the Postgres project was a generation of highly trained DBMS implementers, who have gone on to be instrumental in building several other commercial systems

The third system in this section is Gamma, built at Wisconsin between 1984 and 1990. In my opinion, Gamma popularized the shared-nothing partitioned table approach to multi-node data management. Although Teradata had the same ideas in parallel, it was Gamma that popularized the concepts. In addition, prior to Gamma, nobody talked about hash-joins so Gamma should be credited (along with Kitsuregawa Masaru) with coming up with this class of algorithms.

Essentially all data warehouse systems use a Gamma-style architecture. Any thought of using a shared disk or shared memory system have all but disappeared. Unless network latency and bandwidth get to be comparable to disk bandwidth, I expect the current shared-nothing architecture to continue.

References

[1] D. D. Chamberlin. Early history of sql. *Annals of the History of Computing, IEEE*, 34(4):78–82, 2012.

[2] D. D. Chamberlin and R. F. Boyce. Sequel: A structured english query language. In *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) workshop on Data description, access and control*, pages 249–264. ACM, 1974.

[3] C. J. Date. An architecture for high-level language database extensions. In *SIGMOD*, 1976.

[4] C. J. Date. A critique of the SQL database language. *ACM SIGMOD Record*, 14(3), Nov. 1984.

[5] D. H. Hansson et al. Ruby on rails. `http://www.rubyonrails.org`.

[6] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman. *The design and implementation of the 4.4 BSD operating system*. Pearson Education, 1996.

[7] E. Meijer, B. Beckman, and G. Bierman. Linq: reconciling object, relations and XML in the .NET framework. In *SIGMOD*, 2006.

[8] L. A. Rowe and K. A. Shoens. Data abstraction, views and updates in RIGEL. In *SIGMOD*, 1979.

[9] J. W. Schmidt. Some high level language constructs for data of type relation. *ACM Trans. Database Syst.*, 2(3), Sept. 1977.

[10] M. Stonebraker. The land sharks are on the squawk box. *Communications of the ACM*. To appear.

[11] M. Stonebraker, G. Held, E. Wong, and P. Kreps. The design and implementation of ingres. *ACM Transactions on Database Systems (TODS)*, 1(3):189–222, 1976.