Chapter 4: New DBMS Architectures

Introduced by Michael Stonebraker

Selected Readings:

Mike Stonebraker, Daniel J. Abadi, Adam Batkin, Xuedong Chen, Mitch Cherniack, Miguel Ferreira, Edmond Lau, Amerson Lin, Sam Madden, Elizabeth O'Neil, Pat O'Neil, Alex Rasin, Nga Tran, Stan Zdonik. C-store: A Column-oriented DBMS. *SIGMOD*, 2005.

Cristian Diaconu, Craig Freedman, Erik Ismert, Per-Ake Larson, Pravin Mittal, Ryan Stonecipher, Nitin Verma, Mike Zwilling. Hekaton: SQL Server's Memory-optimized OLTP Engine. *SIGMOD*, 2013.

Stavros Harizopoulos, Daniel J. Abadi, Samuel Madden, Michael Stonebraker. OLTP Through the Looking Glass, and What We Found There. *SIGMOD*, 2008.

Probably the most important thing that has happened in the DBMS landscape is the death of "one size fits all". Until the early 2000's the traditional disk-based row-store architecture was omni-present. In effect, the commercial vendors had a hammer and everything was a nail.

In the last fifteen years, there have been several major upheavals, which we discuss in turn.

First, the community realized that column stores are dramatically superior to row stores in the data warehouse marketplace. Data warehouses found early acceptance in customer facing retail environments and quickly spread to customer facing data in general. Warehouses recorded historical information on customer transactions. In effect, this is the who-what-why-when-where of each customer interaction.

The conventional wisdom is to structure a data warehouse around a central Fact table in which this transactional information is recorded. Surrounding this are dimension tables which record information that can be factored out of the Fact table. In a retail scenario one has dimension tables for Stores, Customers, Products and Time. The result is a so-called star schema [3]. If stores are grouped into regions, then there may be multiple levels of dimension tables and a snowflake schema results.

The key observation is that Fact tables are generally "fat" and often contain a hundred or more attributes. Obviously, they also "long" since there are many, many facts to record. In general, the queries to a data warehouse are a mix of repeated requests (produce a monthy sales report by store) and "ad hoc" ones. In a retail warehouse, for example, one might want to know what is selling in the Northeast when a snowstorm occurs and what is selling along the Atlantic seaboard during hurricanes.

Moreover, nobody runs a select * query to fetch all of the rows in the Fact table. Instead, they invariably specify an aggregate, which retrieves a half-dozen attributes out of the 100 in the table. The next query retrieves a different set, and there is little-to-no locality among the filtering criteria.

In this use case, it is clear that a column store will move a factor of 16 less data from the disk to main memory than a row store will (6 columns versus 100). Hence, it has an unfair advantage. Furthermore, consider a storage block. In a column store, there is a single attribute on that block, while a row store will have 100. Compression will clearly work better on one attribute than on 100. In addition, row stores have a header on the front of each record (in SQLServer it is apparently 16 bytes). In contrast, column stores are careful to have no such header.

Lastly, a row-based executor has an inner loop whereby a record is examined for validity in the output. Hence, the overhead of the inner loop, which is considerable, is paid per record examined. In contrast, the fundamental operation of a column store is to retrieve a column and pick out the qualifying items. As such, the inner-loop overhead is paid once per column examined and not once per row examined. As such a column executor is way more efficient in CPU time and retrieves way less data from the disk. In most real-world environments, column stores are 50–100 times faster than row stores.

Early column stores included Sybase IQ [5], which appeared in the 1990s, and MonetDB [2]. However, the technology dates to the 1970s [1, 4]. In the 2000's C-Store/Vertica appeared as well-financed startup with a high performance implementation. Over the next decade the entire data warehouse market morphed from a row-store world to a column store world. Arguably, Sybase IQ could have done the same thing somewhat earlier, if Sybase had invested more aggressively in the technology and done a multi-node implementation. The advantages of a column executor are persuasively discussed in [2], although it is "down in the weeds" and hard to read.

The second major sea change was the precipitous decline in main memory prices. At the present time, one can buy a 1Terabyte for perhaps $25,000, and a high performance computing cluster with a few terabytes can be bought for maybe $100K. The key insight is that OLTP data bases are just not that big. One terabyte is a very big OLTP data base, and is a candidate for main memory deployment. As noted in the looking glass paper in this section, one does not want to run a disk-based row store when data fits in main memory the overhead is just way too high.

In effect, the OLTP marketplace is now becoming a main memory DBMS marketplace. Again, traditional disk-based row stores are just not competitive. To work well, new solutions are needed for concurrency control, crash recovery, and multi-threading, and I expect OLTP architectures to evolve over the next few years.

My current best guess is that nobody will use traditional two phase locking. Techniques based on timestamp ordering or multiple versions are likely to prevail. The third paper in this section discusses Hekaton, which implements a state-of-the art MVCC scheme.

Crash recovery must also be dealt with. In general, the solution proposed is replication, and on-line failover, which was pioneered by Tandem two decades ago. The traditional wisdom is to write a log, move the log over the network, and then roll forward at the backup site. This active-passive architecture has been shown in [6] to be a factor of 3 inferior to an active-active scheme where the transactions is simply run at each replica. If one runs an active-active scheme, then one must ensure that transactions are run in the same order at each replica. Unfortunately, MVCC does not do this. This has led to interest in deterministic concurrency control schemes, which are likely to be wildly faster in an end-to-end system that MVCC.

In any case, OLTP is going to move to main memory deployment, and a new class of main memory DBMSs is unfolding to support this use case.

The third phenomenon that has unfolded is the “no SQL” movement. In essence, there are 100 or so DBMSs, which support a variety of data models and have the following two characteristics:

1. “Out of box” experience. They are easy for a programmer to get going and do something productive. RDBMSs, in contrast, are very heavyweight, requiring a schema up front.
2. Support for semi-structured data. If every record can have values for different attributes, then a traditional row store will have very, very wide tuples, and be very sparse, and therefore inefficient.

This is a wake-up call to the commercial vendors to make systems that are easier to use and support semi-structured data types, such as JSON. In general, I expect the No SQL market to merge with the SQL market over time as RDBMSs react to the two points noted above.

The fourth sea change is the emergence of the Hadoop/HDFS/Spark environment, which is discussed in Chapter 6.

References

[1] D. S. Batory. On searching transposed files. *ACM Transactions on Database Systems (TODS)*, 4(4), Dec. 1979.

[2] P. A. Boncz, M. Zukowski, and N. Nes. Monetdb/x100: Hyper-pipelining query execution. In *CIDR*, 2005.

[3] R. Kimball and M. Ross. *The data warehouse toolkit: the complete guide to dimensional modeling*. John Wiley & Sons, 2011.

[4] R. Lorie and A. Symonds. A relational access method for interactive applications. *Courant Computer Science Symposia, Vol. 6: Data Base Systems*, 1971.

[5] R. MacNicol and B. French. Sybase iq multiplex-designed for analytics. In *VLDB*, 2004.

[6] N. Malviya, A. Weisberg, S. Madden, and M. Stonebraker. Rethinking main memory OLTP recovery. In *ICDE*, 2014.