Architecture of a Database System

Architecture of a Database System

Joseph M. Hellerstein

University of California
USA
hellerstein@cs.berkeley.edu

Michael Stonebraker

Massachusetts Institute of Technology
USA

James Hamilton

Microsoft Research
USA

now

the essence of knowledge

Boston – Delft

Foundations and Trends® in Databases

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is J. M. Hellerstein, M. Stonebraker and J. Hamilton, Architecture of a Database System, Foundation and Trends® in Databases, vol 1, no 2, pp 141–259, 2007

ISBN: 978-1-60198-078-6

Foundations and Trends® in Databases
Volume 1 Issue 2, 2007

Editorial Board

Full text available at: http://dx.doi.org/10.1561/1900000002

Editorial Scope

Foundations and Trends® in Databases covers a breadth of topics relating to the management of large volumes of data. The journal targets the full scope of issues in data management, from theoretical foundations, to languages and modeling, to algorithms, system architecture, and applications. The list of topics below illustrates some of the intended coverage, though it is by no means exhaustive:

- Data Models and Query Languages
- Query Processing and Optimization
- Storage, Access Methods, and Indexing
- Transaction Management, Concurrency Control and Recovery
- Deductive Databases
- Parallel and Distributed Database Systems
- Database Design and Tuning
- Metadata Management
- Object Management
- Trigger Processing and Active Databases
- Data Mining and OLAP
- Approximate and Interactive Query Processing
- Data Warehousing
- Adaptive Query Processing
- Data Stream Management
- Search and Query Integration
- XML and Semi-Structured Data
- Web Services and Middleware
- Data Integration and Exchange
- Private and Secure Data Management
- Peer-to-Peer, Sensornet and Mobile Data Management
- Scientific and Spatial Data Management
- Data Brokering and Publish/Subscribe
- Data Cleaning and Information Extraction
- Probabilistic Data Management

Information for Librarians
Foundations and Trends® in Databases, 2007, Volume 1, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.

Foundations and Trends® in
Databases
Vol. 1, No. 2 (2007) 141–259

DOI: 10.1561/1900000002

now
the essence of knowledge

Architecture of a Database System

Joseph M. Hellerstein[1], Michael Stonebraker[2] and James Hamilton[3]

[1] *University of California, Berkeley, USA, hellerstein@cs.berkeley.edu*
[2] *Massachusetts Institute of Technology, USA*
[3] *Microsoft Research, USA*

Abstract

Database Management Systems (DBMSs) are a ubiquitous and critical component of modern computing, and the result of decades of research and development in both academia and industry. Historically, DBMSs were among the earliest multi-user server systems to be developed, and thus pioneered many systems design techniques for scalability and reliability now in use in many other contexts. While many of the algorithms and abstractions used by a DBMS are textbook material, there has been relatively sparse coverage in the literature of the systems design issues that make a DBMS work. This paper presents an architectural discussion of DBMS design principles, including process models, parallel architecture, storage system design, transaction system implementation, query processor and optimizer architectures, and typical shared components and utilities. Successful commercial and open-source systems are used as points of reference, particularly when multiple alternative designs have been adopted by different groups.

Contents

1

Introduction

Database Management Systems (DBMSs) are complex, mission-critical software systems. Today's DBMSs embody decades of academic and industrial research and intense corporate software development. Database systems were among the earliest widely deployed online server systems and, as such, have pioneered design solutions spanning not only data management, but also applications, operating systems, and networked services. The early DBMSs are among the most influential software systems in computer science, and the ideas and implementation issues pioneered for DBMSs are widely copied and reinvented.

For a number of reasons, the lessons of database systems architecture are not as broadly known as they should be. First, the applied database systems community is fairly small. Since market forces only support a few competitors at the high end, only a handful of successful DBMS implementations exist. The community of people involved in designing and implementing database systems is tight: many attended the same schools, worked on the same influential research projects, and collaborated on the same commercial products. Second, academic treatment of database systems often ignores architectural issues. Textbook presentations of database systems traditionally focus on algorithmic

and theoretical issues — which are natural to teach, study, and test — without a holistic discussion of system architecture in full implementations. In sum, much conventional wisdom about how to build database systems is available, but little of it has been written down or communicated broadly.

In this paper, we attempt to capture the main architectural aspects of modern database systems, with a discussion of advanced topics. Some of these appear in the literature, and we provide references where appropriate. Other issues are buried in product manuals, and some are simply part of the oral tradition of the community. Where applicable, we use commercial and open-source systems as examples of the various architectural forms discussed. Space prevents, however, the enumeration of the exceptions and finer nuances that have found their way into these multi-million line code bases, most of which are well over a decade old. Our goal here is to focus on overall system design and stress issues not typically discussed in textbooks, providing useful context for more widely known algorithms and concepts. We assume that the reader is familiar with textbook database systems material (e.g., [72] or [83]) and with the basic facilities of modern operating systems such as UNIX, Linux, or Windows. After introducing the high-level architecture of a DBMS in the next section, we provide a number of references to background reading on each of the components in Section 1.2.

1.1 Relational Systems: The Life of a Query

The most mature and widely used database systems in production today are relational database management systems (RDBMSs). These systems can be found at the core of much of the world's application infrastructure including e-commerce, medical records, billing, human resources, payroll, customer relationship management and supply chain management, to name a few. The advent of web-based commerce and community-oriented sites has only increased the volume and breadth of their use. Relational systems serve as the repositories of record behind nearly all online transactions and most online content management systems (blogs, wikis, social networks, and the like). In addition to being important software infrastructure, relational database systems serve as

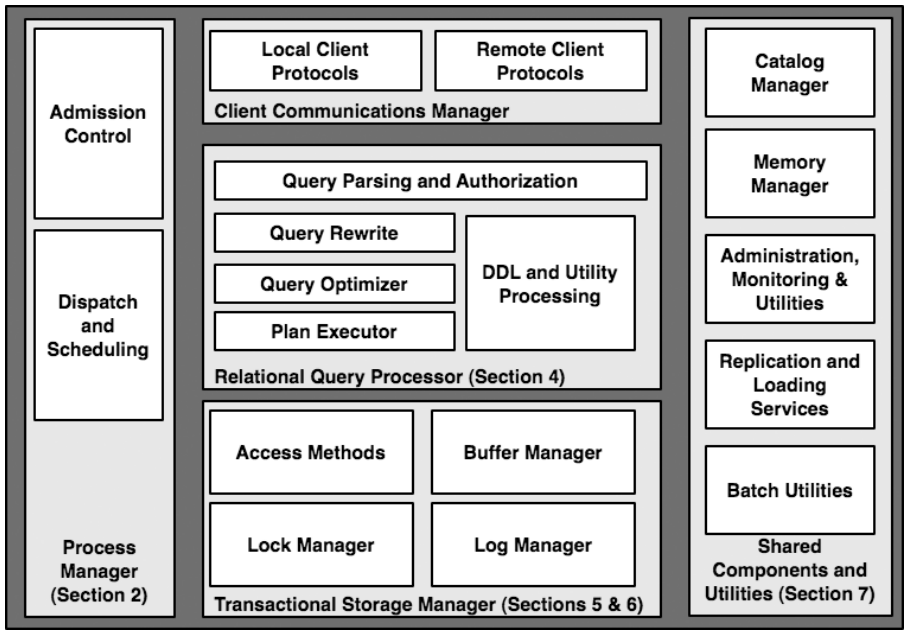

Fig. 1.1 Main components of a DBMS.

a well-understood point of reference for new extensions and revolutions in database systems that may arise in the future. As a result, we focus on relational database systems throughout this paper.

At heart, a typical RDBMS has five main components, as illustrated in Figure 1.1. As an introduction to each of these components and the way they fit together, we step through the life of a query in a database system. This also serves as an overview of the remaining sections of the paper.

Consider a simple but typical database interaction at an airport, in which a gate agent clicks on a form to request the passenger list for a flight. This button click results in a single-query transaction that works roughly as follows:

1. The personal computer at the airport gate (the "client") calls an API that in turn communicates over a network to establish a connection with the *Client Communications Manager* of a DBMS (top of Figure 1.1). In some cases, this connection

is established between the client and the database server directly, e.g., via the ODBC or JDBC connectivity protocol. This arrangement is termed a "two-tier" or "client-server" system. In other cases, the client may communicate with a "middle-tier server" (a web server, transaction processing monitor, or the like), which in turn uses a protocol to proxy the communication between the client and the DBMS. This is usually called a "three-tier" system. In many web-based scenarios there is yet another "application server" tier between the web server and the DBMS, resulting in four tiers. Given these various options, a typical DBMS needs to be compatible with many different connectivity protocols used by various client drivers and middleware systems. At base, however, the responsibility of the DBMS' client communications manager in all these protocols is roughly the same: to establish and remember the connection state for the caller (be it a client or a middleware server), to respond to SQL commands from the caller, and to return both data and control messages (result codes, errors, etc.) as appropriate. In our simple example, the communications manager would establish the security credentials of the client, set up state to remember the details of the new connection and the current SQL command across calls, and forward the client's first request deeper into the DBMS to be processed.

2. Upon receiving the client's first SQL command, the DBMS must assign a "thread of computation" to the command. It must also make sure that the thread's data and control outputs are connected via the communications manager to the client. These tasks are the job of the DBMS *Process Manager* (left side of Figure 1.1). The most important decision that the DBMS needs to make at this stage in the query regards *admission control*: whether the system should begin processing the query immediately, or defer execution until a time when enough system resources are available to devote to this query. We discuss Process Management in detail in Section 2.

3. Once admitted and allocated as a thread of control, the gate agent's query can begin to execute. It does so by invoking the code in the *Relational Query Processor* (center, Figure 1.1). This set of modules checks that the user is authorized to run the query, and compiles the user's SQL query text into an internal *query plan*. Once compiled, the resulting query plan is handled via the plan executor. The plan executor consists of a suite of "operators" (relational algorithm implementations) for executing any query. Typical operators implement relational query processing tasks including joins, selection, projection, aggregation, sorting and so on, as well as calls to request data records from lower layers of the system. In our example query, a small subset of these operators — as assembled by the query optimization process — is invoked to satisfy the gate agent's query. We discuss the query processor in Section 4.
4. At the base of the gate agent's query plan, one or more operators exist to request data from the database. These operators make calls to fetch data from the DBMS' *Transactional Storage Manager* (Figure 1.1, bottom), which manages all data access (read) and manipulation (create, update, delete) calls. The storage system includes algorithms and data structures for organizing and accessing data on disk ("access methods"), including basic structures like tables and indexes. It also includes a buffer management module that decides when and what data to transfer between disk and memory buffers. Returning to our example, in the course of accessing data in the access methods, the gate agent's query must invoke the transaction management code to ensure the well-known "ACID" properties of transactions [30] (discussed in more detail in Section 5.1). Before accessing data, locks are acquired from a lock manager to ensure correct execution in the face of other concurrent queries. If the gate agent's query involved updates to the database, it would interact with the log manager to ensure that the transaction was durable if committed, and fully undone if aborted.

In Section 5, we discuss storage and buffer management in more detail; Section 6 covers the transactional consistency architecture.

5. At this point in the example query's life, it has begun to access data records, and is ready to use them to compute results for the client. This is done by "unwinding the stack" of activities we described up to this point. The access methods return control to the query executor's operators, which orchestrate the computation of result tuples from database data; as result tuples are generated, they are placed in a buffer for the client communications manager, which ships the results back to the caller. For large result sets, the client typically will make additional calls to fetch more data incrementally from the query, resulting in multiple iterations through the communications manager, query executor, and storage manager. In our simple example, at the end of the query the transaction is completed and the connection closed; this results in the transaction manager cleaning up state for the transaction, the process manager freeing any control structures for the query, and the communications manager cleaning up communication state for the connection.

Our discussion of this example query touches on many of the key components in an RDBMS, but not all of them. The right-hand side of Figure 1.1 depicts a number of shared components and utilities that are vital to the operation of a full-function DBMS. The catalog and memory managers are invoked as utilities during any transaction, including our example query. The catalog is used by the query processor during authentication, parsing, and query optimization. The memory manager is used throughout the DBMS whenever memory needs to be dynamically allocated or deallocated. The remaining modules listed in the rightmost box of Figure 1.1 are utilities that run independently of any particular query, keeping the database as a whole well-tuned and reliable. We discuss these shared components and utilities in Section 7.

1.2 Scope and Overview

In most of this paper, our focus is on architectural fundamentals supporting core database functionality. We do not attempt to provide a comprehensive review of database algorithmics that have been extensively documented in the literature. We also provide only minimal discussion of many extensions present in modern DBMSs, most of which provide features beyond core data management but do not significantly alter the system architecture. However, within the various sections of this paper we note topics of interest that are beyond the scope of the paper, and where possible we provide pointers to additional reading.

We begin our discussion with an investigation of the overall architecture of database systems. The first topic in any server system architecture is its overall process structure, and we explore a variety of viable alternatives on this front, first for uniprocessor machines and then for the variety of parallel architectures available today. This discussion of core server system architecture is applicable to a variety of systems, but was to a large degree pioneered in DBMS design. Following this, we begin on the more domain-specific components of a DBMS. We start with a single query's view of the system, focusing on the relational query processor. Following that, we move into the storage architecture and transactional storage management design. Finally, we present some of the shared components and utilities that exist in most DBMSs, but are rarely discussed in textbooks.

References

[1] A. Adya, B. Liskov, and P. O'Neil, "Generalized isolation level definitions," in *16th International Conference on Data Engineering (ICDE)*, San Diego, CA, February 2000.

[2] R. Agrawal, M. J. Carey, and M. Livny, "Concurrency control performance modelling: Alternatives and implications," *ACM Transactions on Database Systems (TODS)*, vol. 12, pp. 609–654, 1987.

[3] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. Gray, P. P. Griffiths, W. F. Frank King III, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson, "System R: Relational approach to database management," *ACM Transactions on Database Systems (TODS)*, vol. 1, pp. 97–137, 1976.

[4] R. Bayer and M. Schkolnick, "Concurrency of operations on B-trees," *Acta Informatica*, vol. 9, pp. 1–21, 1977.

[5] K. P. Bennett, M. C. Ferris, and Y. E. Ioannidis, "A genetic algorithm for database query optimization," in *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 400–407, San Diego, CA, July 1991.

[6] H. Berenson, P. A. Bernstein, J. Gray, J. Melton, E. J. O'Neil, and P. E. O'Neil, "A critique of ANSI SQL isolation levels," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 1–10, San Jose, CA, May 1995.

[7] P. A. Bernstein and N. Goodman, "Concurrency control in distributed database systems," *ACM Computing Surveys*, vol. 13, 1981.

[8] W. Bridge, A. Joshi, M. Keihl, T. Lahiri, J. Loaiza, and N. MacNaughton, "The oracle universal server buffer," in *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB)*, pp. 590–594, Athens, Greece, August 1997.

[9] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," in *Symposium on Operating System Design and Implementation (OSDI)*, 2006.

[10] S. Chaudhuri, "An overview of query optimization in relational systems," in *Proceedings of ACM Principles of Database Systems (PODS)*, 1998.

[11] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," *ACM SIGMOD Record*, March 1997.

[12] S. Chaudhuri and V. R. Narasayya, "Autoadmin 'what-if' index analysis utility," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 367–378, Seattle, WA, June 1998.

[13] S. Chaudhuri and K. Shim, "Optimization of queries with user-defined predicates," *ACM Transactions on Database Systems (TODS)*, vol. 24, pp. 177–228, 1999.

[14] M.-S. Chen, J. Hun, and P. S. Yu, "Data mining: An overview from a database perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, 1996.

[15] H.-T. Chou and D. J. DeWitt, "An evaluation of buffer management strategies for relational database systems," in *Proceedings of 11th International Conference on Very Large Data Bases (VLDB)*, pp. 127–141, Stockholm, Sweden, August 1985.

[16] A. Desphande, M. Garofalakis, and R. Rastogi, "Independence is good: Dependency-based histogram synopses for high-dimensional data," in *Proceedings of the 18th International Conference on Data Engineering*, San Jose, CA, February 2001.

[17] P. Flajolet and G. Nigel Martin, "Probabilistic counting algorithms for data base applications," *Journal of Computing System Science*, vol. 31, pp. 182–209, 1985.

[18] C. A. Galindo-Legaria, A. Pellenkoft, and M. L. Kersten, "Fast, randomized join-order selection — why use transformations?," *VLDB*, pp. 85–95, 1994.

[19] S. Ganguly, W. Hasan, and R. Krishnamurthy, "Query optimization for parallel execution," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 9–18, San Diego, CA, June 1992.

[20] M. Garofalakis and P. B. Gibbons, "Approximate query processing: Taming the terabytes, a tutorial," in *International Conferenence on Very Large Data Bases*, 2001. www.vldb.org/conf/2001/tut4.pdf.

[21] M. N. Garofalakis and Y. E. Ioannidis, "Parallel query scheduling and optimization with time- and space-shared resources," in *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB)*, pp. 296–305, Athens, Greece, August 1997.

[22] R. Goldman and J. Widom, "Wsq/dsq: A practical approach for combined querying of databases and the web," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, 2000.

[23] G. Graefe, "Encapsulation of parallelism in the volcano query processing system," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pp. 102–111, Atlantic City, May 1990.

[24] G. Graefe, "Query evaluation techniques for large databases," *Computing Surveys*, vol. 25, pp. 73–170, 1993.

[25] G. Graefe, "The cascades framework for query optimization," *IEEE Data Engineering Bulletin*, vol. 18, pp. 19–29, 1995.

[26] C. Graham, "Market share: Relational database management systems by operating system, worldwide, 2005," Gartner Report No: G00141017, May 2006.

[27] J. Gray, "Greetings from a filesystem user," in *Proceedings of the FAST '05 Conference on File and Storage Technologies*, (San Francisco), December 2005.

[28] J. Gray and B. Fitzgerald, *FLASH Disk Opportunity for Server-Applications*. http://research.microsoft.com/∼Gray/papers/FlashDiskPublic.doc.

[29] J. Gray, R. A. Lorie, G. R. Putzolu, and I. L. Traiger, "Granularity of locks and degrees of consistency in a shared data base," in *IFIP Working Conference on Modelling in Data Base Management Systems*, pp. 365–394, 1976.

[30] J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

[31] S. D. Gribble, E. A. Brewer, J. M. Hellerstein, and D. Culler, "Scalable, distributed data structures for internet service construction," in *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI)*, 2000.

[32] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pp. 47–57, Boston, June 1984.

[33] L. Haas, D. Kossmann, E. L. Wimmers, and J. Yang, "Optimizing queries across diverse data sources," in *International Conference on Very Large Databases (VLDB)*, 1997.

[34] T. Haerder and A. Reuter, "Principles of transaction-oriented database recovery," *ACM Computing Surveys*, vol. 15, pp. 287–317, 1983.

[35] S. Harizopoulos and N. Ailamaki, "StagedDB: Designing database servers for modern hardware," *IEEE Data Engineering Bulletin*, vol. 28, pp. 11–16, June 2005.

[36] S. Harizopoulos, V. Liang, D. Abadi, and S. Madden, "Performance tradeoffs in read-optimized databases," in *Proceedings of the 32nd Very Large Databases Conference (VLDB)*, 2006.

[37] J. M. Hellerstein, "Optimization techniques for queries with expensive methods," *ACM Transactions on Database Systems (TODS)*, vol. 23, pp. 113–157, 1998.

[38] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online aggregation," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, 1997.

[39] J. M. Hellerstein, J. Naughton, and A. Pfeffer, "Generalized search trees for database system," in *Proceedings of Very Large Data Bases Conference (VLDB)*, 1995.

[40] J. M. Hellerstein and A. Pfeffer, "The russian-doll tree, an index structure for sets," University of Wisconsin Technical Report TR1252, 1994.

[41] C. Hoare, "Monitors: An operating system structuring concept," *Communications of the ACM (CACM)*, vol. 17, pp. 549–557, 1974.

[42] W. Hong and M. Stonebraker, "Optimization of parallel query execution plans in xprs," in *Proceedings of the First International Conference on Parallel and Distributed Information Systems (PDIS)*, pp. 218–225, Miami Beach, FL, December 1991.

[43] H.-I. Hsiao and D. J. DeWitt, "Chained declustering: A new availability strategy for multiprocessor database machines," in *Proceedings of Sixth International Conference on Data Engineering (ICDE)*, pp. 456–465, Los Angeles, CA, November 1990.

[44] Y. E. Ioannidis and Y. Cha Kang, "Randomized algorithms for optimizing large join queries," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pp. 312–321, Atlantic City, May 1990.

[45] Y. E. Ioannidis and S. Christodoulakis, "On the propagation of errors in the size of join results," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 268–277, Denver, CO, May 1991.

[46] M. Kornacker, C. Mohan, and J. M. Hellerstein, "Concurrency and recovery in generalized search trees," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 62–72, Tucson, AZ, May 1997.

[47] H. T. Kung and J. T. Robinson, "On optimistic methods for concurrency control," *ACM Tranactions on Database Systems (TODS)*, vol. 6, pp. 213–226, 1981.

[48] J. R. Larus and M. Parkes, "Using cohort scheduling to enhance server performance," in *USENIX Annual Conference*, 2002.

[49] H. C. Lauer and R. M. Needham, "On the duality of operating system structures," *ACM SIGOPS Operating Systems Review*, vol. 13, pp. 3–19, April 1979.

[50] P. L. Lehman and S. Bing Yao, "Efficient locking for concurrent operations on b-trees," *ACM Transactions on Database Systems (TODS)*, vol. 6, pp. 650–670, December 1981.

[51] A. Y. Levy, "Answering queries using views," *VLDB Journal*, vol. 10, pp. 270–294, 2001.

[52] A. Y. Levy, I. Singh Mumick, and Y. Sagiv, "Query optimization by predicate move-around," in *Proceedings of 20th International Conference on Very Large Data Bases*, pp. 96–107, Santiago, September 1994.

[53] W. Litwin, "Linear hashing: A new tool for file and table addressing," in *Sixth International Conference on Very Large Data Bases (VLDB)*, pp. 212–223, Montreal, Quebec, Canada, October 1980.

[54] G. M. Lohman, "Grammar-like functional rules for representing query optimization alternatives," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 18–27, Chicago, IL, June 1988.

[55] Q. Luo, S. Krishnamurthy, C. Mohan, H. Pirahesh, H. Woo, B. G. Lindsay, and J. F. Naughton, "Middle-tier database caching for e-business," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, 2002.

[56] S. R. Madden and M. J. Franklin, "Fjording the stream: An architecture for queries over streaming sensor data," in *Proceedings of 12th IEEE International Conference on Data Engineering (ICDE)*, San Jose, February 2002.

[57] V. Markl, G. Lohman, and V. Raman, "Leo: An autonomic query optimizer for db2," *IBM Systems Journal*, vol. 42, pp. 98–106, 2003.

[58] C. Mohan, "Aries/kvl: A key-value locking method for concurrency control of multiaction transactions operating on b-tree indexes," in *16th International Conference on Very Large Data Bases (VLDB)*, pp. 392–405, Brisbane, Queensland, Australia, August 1990.

[59] C. Mohan, D. J. Haderle, B. G. Lindsay, H. Pirahesh, and P. M. Schwarz, "Aries: A transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging," *ACM Transactions on Database Systems (TODS)*, vol. 17, pp. 94–162, 1992.

[60] C. Mohan and F. Levine, "Aries/im: An efficient and high concurrency index management method using write-ahead logging," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, (M. Stonebraker, ed.), pp. 371–380, San Diego, CA, June 1992.

[61] C. Mohan, B. G. Lindsay, and R. Obermarck, "Transaction management in the r* distributed database management system," *ACM Transactions on Database Systems (TODS)*, vol. 11, pp. 378–396, 1986.

[62] E. Nightingale, K. Veerarghavan, P. M. Chen, and J. Flinn, "Rethink the sync," in *Symposium on Operating Systems Design and Implementation (OSDI)*, November 2006.

[63] OLAP Market Report. Online manuscript. http://www.olapreport.com/market.htm.

[64] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "The lru-k page replacement algorithm for database disk buffering," in *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 297–306, Washington, DC, May 1993.

[65] P. E. O'Neil and D. Quass, "Improved query performance with variant indexes," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pp. 38–49, Tucson, May 1997.

[66] S. Padmanabhan, B. Bhattacharjee, T. Malkemus, L. Cranston, and M. Huras, "Multi-dimensional clustering: A new data layout scheme in db2," in *ACM SIGMOD International Management of Data* (San Diego, California, June 09–12, 2003) *SIGMOD '03*, pp. 637–641, New York, NY: ACM Press, 2003.

[67] D. Patterson, "Latency lags bandwidth," *CACM*, vol. 47, pp. 71–75, October 2004.

[68] H. Pirahesh, J. M. Hellerstein, and W. Hasan, "Extensible/rule- based query rewrite optimization in starburst," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pp. 39–48, San Diego, June 1992.

[69] V. Poosala and Y. E. Ioannidis, "Selectivity estimation without the attribute value independence assumption," in *Proceedings of 23rd International Conference on Very Large Data Bases (VLDB)*, pp. 486–495, Athens, Greece, August 1997.

[70] M. Pöss, B. Smith, L. Kollár, and P.-Å. Larson, "Tpc-ds, taking decision support benchmarking to the next level," in *SIGMOD 2002*, pp. 582–587.

[71] V. Prabakharan, A. C. Arpaci-Dusseau, and R. Arpaci-Dusseau, "Analysis and evolution of journaling file systems," in *Proceedings of USENIX Annual Technical Conference*, April 2005.

[72] R. Ramakrishnan and J. Gehrke, "Database management systems," McGraw-Hill, Boston, MA, Third ed., 2003.

[73] V. Raman and G. Swart, "How to wring a table dry: Entropy compression of relations and querying of compressed relations," in *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 2006.

[74] D. P. Reed, *Naming and Synchronization in a Decentralized Computer System*. PhD thesis, MIT, Dept. of Electrical Engineering, 1978.

[75] A. Reiter, "A study of buffer management policies for data management systems," Technical Summary Report 1619, Mathematics Research Center, University of Wisconsin, Madison, 1976.

[76] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "System level concurrency control for distributed database systems," *ACM Transactions on Database Systems (TODS)*, vol. 3, pp. 178–198, June 1978.

[77] S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating mining with relational database systems: Alternatives and implications," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, 1998.

[78] R. Sears and E. Brewer, "Statis: Flexible transactional storage," in *Proceedings of Symposium on Operating Systems Design and Implementation (OSDI)*, 2006.

[79] P. G. Selinger, M. Astrahan, D. Chamberlin, R. Lorie, and T. Price, "Access path selection in a relational database management system," in *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pp. 22–34, Boston, June 1979.

[80] P. Seshadri, H. Pirahesh, and T. Y. C. Leung, "Complex query decorrelation," in *Proceedings of 12th IEEE International Conference on Data Engineering (ICDE)*, New Orleans, February 1996.

[81] M. A. Shah, S. Madden, M. J. Franklin, and J. M. Hellerstein, "Java support for data-intensive systems: Experiences building the telegraph dataflow system," *ACM SIGMOD Record*, vol. 30, pp. 103–114, 2001.

[82] L. D. Shapiro, "Exploiting upper and lower bounds in top-down query optimization," *International Database Engineering and Application Symposium (IDEAS)*, 2001.

[83] A. Silberschatz, H. F. Korth, and S. Sudarshan, *Database System Concepts*. McGraw-Hill, Boston, MA, Fourth ed., 2001.

[84] M. Steinbrunn, G. Moerkotte, and A. Kemper, "Heuristic and randomized optimization for the join ordering problem," *VLDB Journal*, vol. 6, pp. 191–208, 1997.

[85] M. Stonebraker, "Retrospection on a database system," *ACM Transactions on Database Systems (TODS)*, vol. 5, pp. 225–240, 1980.

[86] M. Stonebraker, "Operating system support for database management," *Communications of the ACM (CACM)*, vol. 24, pp. 412–418, 1981.

[87] M. Stonebraker, "The case for shared nothing," *IEEE Database Engineering Bulletin*, vol. 9, pp. 4–9, 1986.

[88] M. Stonebraker, "Inclusion of new types in relational data base systems," *ICDE*, pp. 262–269, 1986.

[89] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. O'Neil, P. O'Neil, A. Rasin, N. Tran,

and S. Zdonik, "C-store: A column oriented dbms," in *Proceedings of the Conference on Very Large Databases (VLDB)*, 2005.

[90] M. Stonebraker and U. Cetintemel, "One size fits all: An idea whose time has come and gone," in *Proceedings of the International Conference on Data Engineering (ICDE)*, 2005.

[91] Transaction Processing Performance Council 2006. TPC Benchmark C Standard Specification Revision 5.7, http://www.tpc.org/tpcc/spec/tpcc_current.pdf, April.

[92] T. Urhan, M. J. Franklin, and L. Amsaleg, "Cost based query scrambling for initial delays," *ACM-SIGMOD International Conference on Management of Data*, 1998.

[93] R. von Behren, J. Condit, F. Zhou, G. C. Necula, and E. Brewer, "Capriccio: Scalable threads for internet services," in *Proceedings of the Ninteenth Symposium on Operating System Principles (SOSP-19)*, Lake George, New York, October 2003.

[94] M. Welsh, D. Culler, and E. Brewer, "Seda: An architecture for well- conditioned, scalable internet services," in *Proceedings of the 18th Symposium on Operating Systems Principles (SOSP-18)*, Banff, Canada, October 2001.

[95] C. Zou and B. Salzberg, "On-line reorganization of sparsely-populated b+trees," pp. 115–124, 1996.