Adaptive Query Processing

Adaptive Query Processing

Amol Deshpande

University of Maryland
USA
amol@cs.umd.edu

Zachary Ives

University of Pennsylvania
USA
zives@cis.upenn.edu

Vijayshankar Raman

IBM Almaden
USA
ravijay@us.ibm.com

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 A. Deshpande, Z. Ives and V. Raman, Adaptive Query Processing, Foundation and Trends® in Databases, vol 1, no 1, pp 1–140, 2007

ISBN: 978-1-60198-034-2

Foundations and Trends® in Databases

Volume 1 Issue 1, 2007

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. 1 (2007) 1–140

DOI: 10.1561/1900000001

Adaptive Query Processing

Amol Deshpande[1], Zachary Ives[2] and Vijayshankar Raman[3]

[1] *University of Maryland, USA, amol@cs.umd.edu*
[2] *University of Pennsylvania, USA, zives@cis.upenn.edu*
[3] *IBM Almaden, USA, ravijay@us.ibm.com*

Abstract

As the data management field has diversified to consider settings in which queries are increasingly complex, statistics are less available, or data is stored remotely, there has been an acknowledgment that the traditional optimize-then-execute paradigm is insufficient. This has led to a plethora of new techniques, generally placed under the common banner of *adaptive query processing*, that focus on using runtime feedback to modify query processing in a way that provides better response time or more efficient CPU utilization.

In this survey paper, we identify many of the common issues, themes, and approaches that pervade this work, and the settings in which each piece of work is most appropriate. Our goal with this paper is to be a "value-add" over the existing papers on the material, providing not only a brief overview of each technique, but also a basic framework for understanding the field of adaptive query processing in general. We focus primarily on *intra-query* adaptivity of long-running, but not full-fledged streaming, queries. We conclude with a discussion of open research problems that are of high importance.

Contents

1

Introduction

One of the fundamental breakthroughs of Codd's relational data model [33] was the identification of how to take a *declarative*, logic-based formulation of a query and convert it into an algebraic query evaluation tree. As described in every database textbook, this enabled physical data independence and promised many benefits: the database administrator and the DBMS optimizer became free to choose among many different storage formats and execution plans to answer a declarative query. The challenge, since then, has been how to deliver on these promises — regardless of where or how the data is laid out, how complex the query is, and how unpredictable the operating environment is.

This challenge has spurred 30 years of query processing research. Cost-based query optimization, pioneered by Selinger *et al.* [102] in System R and refined by generations of database researchers and developers, has been tremendously effective in addressing the needs of relational DBMS query processing: one can get excellent performance for queries over data with few correlations, executed in a relatively stable environment, given sufficient statistical information.

However, when even one of these characteristics is not present, the System R-style optimize-then-execute model begins to break down: as noted in [69], optimizer error begins to build up at a rate exponential in the size of the query. As the database field has broadened to consider more general *data management*, including querying autonomous remote data sources, supporting continuous queries over data streams, encoding and retrieving XML data, supporting OLAP and data mining operations, and combining text search with structured query capabilities, the weaknesses of the traditional optimization model have begun to show themselves.

In response, there has been a surge of interest in a broad array of techniques termed *adaptive query processing* (AQP). AQP addresses the problems of missing statistics, unexpected correlations, unpredictable costs, and dynamic data by using *feedback* to *tune* execution. It is one of the cornerstones of so-called *autonomic* database management systems, although it also generalizes to many other contexts, particularly at the intersection of database query processing and the Web.

The spectrum of adaptive query processing techniques has been quite broad: they may span multiple query executions or adapt within the execution of a single query; they may affect the query plan being executed or the scheduling of operations within the plan; they have been developed for improving performance of local DBMS queries (e.g., [75, 87, 112]), for processing distributed and streaming data (e.g., [6, 72, 88, 92, 101]), and for performing distributed query execution (e.g., [115]).

This survey is an attempt to cover the fundamental issues, techniques, costs, and benefits of adaptive query processing. We begin with a broad overview of the field, identifying the dimensions of adaptive techniques. Then we focus our analysis on the spectrum of approaches available to adapt query execution at runtime — primarily in a non-streaming context. Where possible, we focus on simplifying and abstracting the key concepts of each technique, rather than reproducing the full details available in the papers; we consider generalizations of the specific published implementations. Our goal is to identify the strengths and limitations of the different techniques, demonstrate when they are most useful, and suggest possible avenues of future research.

In the rest of the section, we present a brief overview of query processing in relational database systems (Section 1.1) and elaborate on the reasons behind the push toward adaptivity (Section 1.2); we then present a road map for the rest of the survey (Section 1.3), and briefly discuss the related surveys of interest (Section 1.4).

1.1 Query Processing in Relational Database Systems

The conventional method of processing a query in a relational DBMS is to parse the SQL statement and produce a relational calculus-like logical representation of the query, and then to invoke the query optimizer, which generates a *query plan*. The query plan is fed into an execution engine that directly executes it, typically with little or no runtime decision-making (Figure 1.1).

The query plan can be thought of as a tree of unary and binary relational algebra operators, where each operator is annotated with specific details about the algorithm to use (e.g., nested loops join versus hash join) and how to allocate resources (e.g., memory). In many cases the query plan also includes low-level "physical" operations like sorting, network shipping, etc. that do not affect the logical representation of the data.

Certain query processors consider only restricted types of queries, rather than full-blown SQL. A common example of this is select-project-join or SPJ queries: an SPJ query essentially represents a single SQL `SELECT-FROM-WHERE` block with no aggregation or subqueries.

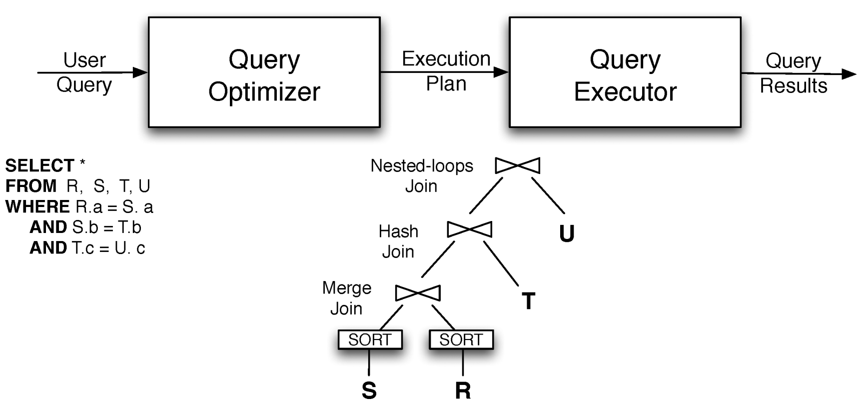

Fig. 1.1 Query processing in database systems.

An even further restriction is *conjunctive* queries, which are SPJ queries that only have conjunctive predicates in the `WHERE` clause; these can be represented as single rules in the Datalog language.

The model of query processing established with the System R project [102], which is still followed today, is to divide query processing into three major stages.

Statistics generation is done offline (typically using the `RUNSTATS` or `UPDATE STATISTICS` command) on the tables in the database. The system profiles the relation instances, collecting information about cardinalities and numbers of unique attribute values, and often generating histograms over certain fields.

The second stage, which is normally done at runtime,[1] is *query optimization*. The optimization stage is very similar to traditional compilation; in fact, in some systems, it generates directly executable code. Optimization uses a combination of *cost estimation*, where the running times of query subexpressions are estimated (based on known performance characteristics of algorithm implementations, calibration parameters for hardware speed, and the statistics generated for the relations), pruning heuristics (which are necessary to reduce the overall search space), and exhaustive enumeration. For relatively simple queries with good statistics, the plans produced by a query optimizer can be quite good, although as discussed previously, this is less true in more complex settings.

The final stage, *query execution*, is handled by an engine analogous to a virtual machine or interpreter for the compiled query plan. There are several important aspects of query execution that are of note. The first is that in general it is desirable to *pipeline* computation, such that each operator processes a tuple at a time from its sub-operators, and also propagates a single tuple to its parent for processing. This leads to better response time in terms of initial answers, and often higher throughput as delays are masked. However, not all operators are naturally amenable to pipelining (e.g., operators like sorting and grouping often must process entire table before they can determine

[1] Except for certain embedded SQL queries, which may be pre-optimized or optimized once for multiple possible input bindings.

what tuple to output next). Also, complex query plans may require too many resources to be fully pipelined. In these settings, the optimizer must break the plan into multiple segments, *materializing* (storing) intermediate results at the end of each stage and using that as an input to the next stage.

Second, the issue of *scheduling* computation in a query plan has many performance implications. Traditional query processing makes the assumption that an individual operator implementation (e.g., a nested loops join) should be able to control how CPU cycles are allocated to its child operators. This is achieved through a so-called *iterator* [53] architecture: each operator has `open`, `close`, and `getNextTuple` methods. The query engine first invokes the query plan root node's `open` method, which in turn `open`s its children, and the process repeats recursively down the plan. Then `getNextTuple` is called on the root node. Depending on the operator implementation, it will make calls to its children's `getNextTuple` methods until it can return a tuple to its parent. The process completes until no more tuples are available, and then the engine `close`s the query plan.

An alternate approach, so called *data-driven* or *dataflow* scheduling [121], is used in many parallel database systems. Here, in order to allow for concurrent computation across many machines, the data producers — not the consumers — control the scheduling. Each operator takes data from an input queue, processes it, and sends it to an output queue. Scheduling is determined by the rates at which the queues are filled and emptied. In this survey, we will discuss a number of adaptive techniques that in essence use a hybrid of the iterator and data-driven approaches.

1.2 Motivations for AQP

Over the years, many refinements have been made to the basic query processing technology discussed above. Since CPUs are more powerful today and query workloads are much more diverse, query optimizers perform a more comprehensive search of the space of query plans with joins, relying less on pruning heuristics. Selectivity estimation techniques have become more accurate and consider skewed distributions (and to a limited extent, attribute correlations). However, the System R-style approach has begun to run into its fundamental limits in recent years, primarily due to the emergence of new application domains in which database query processing is being applied. In particular, triggers for this breakdown include the following:

- **Unreliable cardinality estimates:** The cost estimation process depends critically on estimates of the cardinality of various query subexpressions. Despite significant work on building better statistics structures and data collection schemes, many real-world settings have either inaccurate or missing statistics. (In some circumstances, as with remote data sources, statistics may be difficult or impossible to obtain.) Even when base-table statistics are perfect, correlations between predicates can cause intermediate result cardinality estimates to be off by several orders of magnitude [69, 112].
- **Queries with parameter markers:** SQL is not a pleasant language for end users, so most database queries are issued by a user clicking on a form. The SQL for such queries invariably contains parameter markers (for form input), and the precomputed query plans for such queries can be substantially worse than optimal for some values of the parameters.
- **Dynamically changing data, runtime, and workload characteristics:** In many environments, especially data streams [23, 88, 92], queries might be long-running, and the data characteristics and hence the optimal query plans might change during the execution of the query. The runtime costs can also change dramatically, especially in wide-area environments. Similarly, fluctuating query workloads can result in variations in the resources available to execute a query (e.g., memory), making it necessary to adapt.
- **Complex queries involving many tables:** Query optimizers typically switch to a heuristic approach when queries become too complex to be optimized using the dynamic programming approach. Such queries are naturally more prone

to estimation errors [69], and the use of heuristics exacerbates the problem.

- **Interactive querying:** The optimize-then-execute model does not mesh well with an interactive environment where a user might want to cancel or refine a query after a few seconds of execution: the metric changes too quickly for optimization to pay off [61]. Also, pipelined execution and early-result scheduling, even in the presence of slow data sources, becomes paramount.
- **Need for aggressive sharing:** Though there has been much work in multi-query optimization, so far no definitive solutions have emerged in this area. Traditional databases make do with almost no inter-query state sharing because their usage pattern is made up of a small number of queries against large databases. However, sharing the data as well as the computation is critical in environments such as data streams, which feature a very large number of (typically simple) queries over a small set of data streams [28, 86].

There have been two responses to the challenges posed above. The first, a very pragmatic response by application vendors, has been to build domain-specific optimization capabilities *outside* the DBMS and override its local optimizer. Many commercial DBMSs allow users to specify "hints" on what access methods and join orders to use, via SQL or catalog tables. Recently, SAP has built an application-level query processor that runs only a very limited set of plans (essentially, only table scans), but at very high efficiency [82]. While this achieves SAP's target of satisfying its users, it runs counter to the database community's goals of developing high-performance, general-purpose processors for declarative queries.

Our interest in this survey is on the second development, which has been the focus of the academic and commercial DBMS research community: the design and construction of what have come to be known as *adaptive* (or autonomic) query processing systems, that use runtime feedback to adapt query processing.

1.3 Road Map

We begin with a brief introduction to query optimization in relational database systems (Section 2). We then discuss some of the foundations of AQP, namely, three new operators, and several unifying concepts that we use throughout the survey to illustrate the AQP techniques, to analyze them, and to differentiate between them (Section 3).

We begin our discussion of adaptive query processing by considering a simple class of queries called selection ordering queries (Section 4). The discussion of adaptation techniques for join queries is divided into three parts, roughly based on the space of the query execution plans they explore. We begin with a discussion of techniques for adapting pipelined query execution (Sections 6 and 7), and cover non-pipelined query execution in Section 8. We conclude the survey with a discussion of some of the most important research challenges in adaptive query processing (Section 9).

1.4 Related Work

A number of surveys on query processing are related to this paper. We assume basic familiarity with many of the ideas of Graefe's survey on query execution techniques [53]. Kossmann's survey on distributed query processing [79] also provides useful context for the discussion, as do Ioannidis and Chaudhuri's surveys on query optimization [24, 68]. Babu and Bizarro [8] also present a survey of AQP from a different means of classification from our own (whether the scheme is plan-based, routing-based, or continuous query-based).

References

[1] A. Aboulnaga and S. Chaudhuri, "Self-tuning histograms: building histograms without looking at data," in *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 181–192, ACM Press, 1999.

[2] L. Amsaleg, M. J. Franklin, A. Tomasic, and T. Urhan, "Scrambling query plans to cope with unexpected delays," in *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems,* Miami Beach, FL, pp. 208–219, IEEE Computer Society, December 18–20 1996.

[3] G. Antoshenkov and M. Ziauddin, "Query processing and optimization in Oracle Rdb," *The VLDB Journal*, vol. 5, no. 4, pp. 229–237, 1996.

[4] A. Arasu, S. Babu, and J. Widom, "The CQL continuous query language: Semantic foundations and query execution," *The VLDB Journal*, vol. 15, no. 2, pp. 121–142, 2006.

[5] R. H. Arpaci-Dusseau, E. Anderson, N. Treuhaft, D. E. Culler, J. M. Hellerstein, D. Patterson, and K. Yelick, "Cluster I/O with River: making the fast case common," in *IOPADS '99: Proceedings of the sixth workshop on I/O in parallel and distributed systems*, (New York, NY, USA), pp. 10–22, ACM Press, 1999.

[6] R. Avnur and J. M. Hellerstein, "Eddies: continuously adaptive query processing," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 261–272, ACM Press, 2000.

[7] B. Babcock and S. Chaudhuri, "Towards a robust query optimizer: a principled and practical approach," in *SIGMOD '05: Proceedings of the 2005 ACM*

SIGMOD international conference on Management of data, (New York, NY, USA), pp. 119–130, ACM Press, 2005.

[8] S. Babu and P. Bizarro, "Adaptive query processing in the looking glass," in *CIDR '05: Second Biennial Conference on Innovative Data Systems Research*, pp. 238–249, Asilomar, CA, 2005.

[9] S. Babu, P. Bizarro, and D. DeWitt, "Proactive re-optimization," in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data,* Baltimore, MD, pp. 107–118, New York, NY: ACM Press, June 14–16 2005.

[10] S. Babu, R. Motwani, K. Munagala, I. Nishizawa, and J. Widom, "Adaptive ordering of pipelined stream filters," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 407–418, ACM Press, 2004.

[11] S. Babu, K. Munagala, J. Widom, and R. Motwani, "Adaptive caching for continuous queries," in *ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, (Washington, DC, USA), pp. 118–129, IEEE Computer Society, 2005.

[12] S. Babu and J. Widom, "Continuous queries over data streams," *SIGMOD Rec.*, vol. 30, no. 3, pp. 109–120, 2001.

[13] S. Babu and J. Widom, "StreaMon: an adaptive engine for stream query processing," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 931–932, ACM Press, 2004.

[14] M. Balazinska, H. Balakrishnan, S. Madden, and M. Stonebraker, "Fault-tolerance in the Borealis distributed stream processing system," in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 13–24, ACM Press, 2005.

[15] L. Bellatreche, K. Karlapalem, M. K. Mohania, and M. Schneider, "What can partitioning do for your data warehouses and data marts?," in *IDEAS '00: Proceedings of the 2000 International Symposium on Database Engineering & Applications*, (Washington, DC, USA), pp. 437–446, IEEE Computer Society, 2000.

[16] D. A. Berry and B. Fristedt, *Bandit Problems: Sequential Allocation of Experiments*. Springer, October 1985.

[17] P. Bizarro, S. Babu, D. DeWitt, and J. Widom, "Content-based routing: different plans for different data," in *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pp. 757–768, VLDB Endowment, 2005.

[18] P. Bizarro and D. DeWitt, "Adaptive and robust query processing with SHARP," Tech. Rep. 1562, University of Wisconsin – Madison, CS Dept., 2006.

[19] P. G. Bizarro, *Adaptive Query Processing: Dealing with Incomplete and Uncertain Statistics*. PhD thesis, University of Wisconsin – Madison, 2006.

[20] P. D. Bra and J. Paredaens, "Horizontal decompositions and their impact on query solving," *SIGMOD Rec.*, vol. 13, no. 1, pp. 46–50, 1982.

[21] N. Bruno and S. Chaudhuri, "Exploiting statistics on query expressions for optimization," in *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 263–274, ACM Press, 2002.

[22] N. Bruno, S. Chaudhuri, and L. Gravano, "STHoles: a multidimensional workload-aware histogram," in *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 211–222, ACM Press, 2001.

[23] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah, "TelegraphCQ: Continuous dataflow processing for an uncertain world," in *CIDR '03: First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, 2003.

[24] S. Chaudhuri, "An overview of query optimization in relational systems," in *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, (New York, NY, USA), pp. 34–43, ACM Press, 1998.

[25] S. Chaudhuri, U. Dayal, and T. W. Yan, "Join queries with external text sources: Execution and optimization techniques," in *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pp. 410–422, ACM Press, May 26 1995.

[26] S. Chaudhuri and K. Shim, "Including group-by in query optimization," in *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 354–366, Morgan Kaufmann Publishers Inc., 1994.

[27] C. M. Chen and N. Roussopoulos, "Adaptive selectivity estimation using query feedback," in *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 161–172, ACM Press, 1994.

[28] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: a scalable continuous query system for internet databases," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 379–390, ACM Press, 2000.

[29] H.-T. Chou and D. J. DeWitt, "An evaluation of buffer management strategies for relational database systems," in *VLDB '85: Proceedings of 11th International Conference on Very Large Data Bases*, pp. 127–141, Stockholm, Sweden: Morgan Kaufmann, August 21–23 1985.

[30] F. Chu, J. Halpern, and J. Gehrke, "Least expected cost query optimization: what can we expect?," in *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, (New York, NY, USA), pp. 293–302, ACM Press, 2002.

[31] F. Chu, J. Y. Halpern, and P. Seshadri, "Least expected cost query optimization: an exercise in utility," in *PODS '99: Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, (New York, NY, USA), pp. 138–147, ACM Press, 1999.

[32] J. Claussen, A. Kemper, G. Moerkotte, K. Peithner, and M. Steinbrunn, "Optimization and evaluation of disjunctive queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 2, pp. 238–260, 2000.

[33] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 26, no. 1, pp. 64–69, 1983.

[34] R. L. Cole and G. Graefe, "Optimization of dynamic query evaluation plans," in *SIGMOD '94: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pp. 150–160, Minneapolis, MN: ACM Press, May 24–27 1994.

[35] A. Condon, A. Deshpande, L. Hellerstein, and N. Wu, "Flow algorithms for two pipelined filter ordering problems," in *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, (New York, NY, USA), pp. 193–202, ACM Press, 2006.

[36] D. Daniels, "Query compilation in a distributed database system," Tech. Rep., IBM, 1982. Research Report RJ 3423.

[37] A. Deshpande, "An initial study of overheads of eddies," *SIGMOD Rec.*, vol. 33, no. 1, pp. 44–49, 2004.

[38] A. Deshpande, M. Garofalakis, and R. Rastogi, "Independence is good: dependency-based histogram synopses for high-dimensional data," in *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 199–210, ACM Press, 2001.

[39] A. Deshpande, C. Guestrin, W. Hong, and S. Madden, "Exploiting correlated attributes in acquisitional query processing," in *ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, (Washington, DC, USA), pp. 143–154, IEEE Computer Society, 2005.

[40] A. Deshpande and J. M. Hellerstein, "Lifting the burden of history from adaptive query processing," in *VLDB '04: Proceedings of the 30th International Conference on Very Large Data Bases,* Toronto, Canada, August 29–September 3 2004.

[41] A. Deshpande and L. Hellerstein, "Flow algorithms for parallel query optimization," Tech. Rep. CS-TR-4873, University of Maryland at College Park, 2007.

[42] L. Ding and E. A. Rundensteiner, "Evaluating window joins over punctuated streams," in *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, (New York, NY, USA), pp. 98–107, ACM Press, 2004.

[43] O. Etzioni, S. Hanks, T. Jiang, R. M. Karp, O. Madani, and O. Waarts, "Efficient information gathering on the internet," in *FOCS '96: Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, pp. 234–243, IEEE Computer Society, 1996.

[44] S. Ewen, H. Kache, V. Markl, and V. Raman, "Progressive query optimization for federated queries," in *EDBT '06: Proceedings of the 10th International Conference on Extending Database Technology*, pp. 847–864, 2006.

[45] U. Feige, L. Lovász, and P. Tetali, "Approximating min-sum set cover," *Algorithmica*, vol. 40, no. 4, pp. 219–234, 2004.

[46] S. Ganguly, "Design and analysis of parametric query optimization algorithms," in *VLDB '98: Proceedings of the 24th International Conference on Very Large Data Bases*, pp. 228–238, Morgan Kaufmann, August 24–27 1998.

[47] S. Ganguly, W. Hasan, and R. Krishnamurthy, "Query optimization for parallel execution," in *SIGMOD '92: Proceedings of the 1992 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 9–18, ACM Press, 1992.

[48] K. Gao, S. Harizopoulos, I. Pandis, V. Shkapenyuk, and A. Ailamaki, "Simultaneous pipelining in QPipe: Exploiting work sharing opportunities across queries," in *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, (Washington, DC, USA), p. 162, IEEE Computer Society, 2006.

[49] L. Getoor, B. Taskar, and D. Koller, "Selectivity estimation using probabilistic models," in *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 461–472, ACM Press, 2001.

[50] R. Goldman and J. Widom, "DataGuides: Enabling query formulation and optimization in semistructured databases," in *VLDB '97: Proceedings of 23rd International Conference on Very Large Data Bases*, pp. 436–445, Athens, Greece: Morgan Kaufman, August 25–29 1997.

[51] R. Goldman and J. Widom, "WSQ/DSQ: a practical approach for combined querying of databases and the web," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 285–296, ACM Press, 2000.

[52] G. Graefe and K. Ward, "Dynamic query evaluation plans," in *SIGMOD '89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 358–366, ACM Press, 1989.

[53] G. Graefe, "Query evaluation techniques for large databases," *ACM Comput. Surv.*, vol. 25, no. 2, pp. 73–169, 1993.

[54] G. Graefe, "The Cascades framework for query optimization," *IEEE Data Engineering Bulletin*, vol. 18, no. 3, pp. 19–29, 1995.

[55] G. Graefe, R. Bunker, and S. Cooper, "Hash joins and hash teams in Microsoft SQL Server," in *VLDB '98: Proceedings of 24th International Conference on Very Large Data Bases*, pp. 86–97, Morgan Kaufman, August 24–27 1998.

[56] G. Graefe and W. J. McKenna, "The Volcano optimizer generator: Extensibility and efficient search," in *ICDE '93: Proceedings of the Ninth International Conference on Data Engineering,* Vienna, Austria, pp. 209–218, IEEE Computer Society, April 19–23 1993.

[57] H. Guo, P.-Å. Larson, R. Ramakrishnan, and J. Goldstein, "Relaxed currency and consistency: how to say "good enough" in SQL," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 815–826, ACM Press, 2004.

[58] L. M. Haas, J. C. Freytag, G. M. Lohman, and H. Pirahesh, "Extensible query processing in starburst," in *SIGMOD '89: Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 377–388, ACM Press, 1989.

[59] P. J. Haas and J. M. Hellerstein, "Ripple joins for online aggregation," in *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 287–298, ACM Press, 1999.
[60] J. M. Hellerstein, "Optimization techniques for queries with expensive methods," *ACM Transactions on Database Systems*, vol. 23, no. 2, pp. 113–157, 1998.
[61] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas, "Interactive data analysis: The Control project," *Computer*, vol. 32, no. 8, pp. 51–59, 1999.
[62] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Machine Learning*, vol. 32, pp. 151–178, August 1998.
[63] D. A. Huffman, "A method for the construction of minimum redundancy codes," in *Proc. Inst. Radio Eng.*, pp. 1098–1101, 1952.
[64] A. Hulgeri and S. Sudarshan, "Parametric query optimization for linear and piecewise linear cost functions.," in *VLDB '02: Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pp. 167–178, 2002.
[65] A. Hulgeri and S. Sudarshan, "AniPQO: Almost non-intrusive parametric query optimization for nonlinear cost functions.," in *VLDB '03: Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pp. 766–777, 2003.
[66] J.-H. Hwang, M. Balazinska, A. Rasin, U. Cetintemel, M. Stonebraker, and S. Zdonik, "High-availability algorithms for distributed stream processing," in *ICDE '05: Proceedings of the 21st International Conference on Data Engineering (ICDE'05)*, (Washington, DC, USA), pp. 779–790, IEEE Computer Society, 2005.
[67] T. Ibaraki and T. Kameda, "On the optimal nesting order for computing N-relational joins," *ACM Transactions on Database Systems*, vol. 9, no. 3, pp. 482–502, 1984.
[68] Y. E. Ioannidis, "Query optimization," *ACM Computing Surveys*, vol. 28, no. 1, pp. 121–123, 1996.
[69] Y. E. Ioannidis and S. Christodoulakis, "On the propagation of errors in the size of join results," in *SIGMOD '91: Proceedings of the 1991 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 268–277, ACM Press, 1991.
[70] Y. E. Ioannidis, R. T. Ng, K. Shim, and T. K. Sellis, "Parametric query optimization," *The VLDB Journal*, vol. 6, no. 2, pp. 132–151, 1997.
[71] Z. G. Ives, *Efficient Query Processing for Data Integration*. PhD thesis, University of Washington, August 2002.
[72] Z. G. Ives, D. Florescu, M. Friedman, A. Levy, and D. S. Weld, "An adaptive query execution system for data integration," in *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 299–310, ACM Press, 1999.
[73] Z. G. Ives, A. Y. Halevy, and D. S. Weld, "Adapting to source properties in processing data integration queries," in *SIGMOD '04: Proceedings of the 2004*

ACM SIGMOD international conference on Management of data, (New York, NY, USA), pp. 395–406, ACM Press, 2004.

[74] Z. G. Ives and N. E. Taylor, "Sideways information passing for push-style query processing," Tech. Rep. MS-CIS-07-14, University of Pennsylvania, 2007.

[75] N. Kabra and D. J. DeWitt, "Efficient mid-query re-optimization of sub-optimal query execution plans," in *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 106–117, ACM Press, 1998.

[76] H. Kaplan, E. Kushilevitz, and Y. Mansour, "Learning with attribute costs," in *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 356–365, ACM Press, 2005.

[77] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Machine Learning*, vol. 49, pp. 260–268, November 2002.

[78] W. Kim, "On optimizing an SQL-like nested query," *ACM Transactions on Database Systems*, vol. 7, no. 3, pp. 443–469, 1982.

[79] D. Kossmann, "The state of the art in distributed query processing," *ACM Comput. Surv.*, vol. 32, no. 4, pp. 422–469, 2000.

[80] R. Krishnamurthy, H. Boral, and C. Zaniolo, "Optimization of nonrecursive queries," in *VLDB '86: Proceedings of the 12th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 128–137, Morgan Kaufmann Publishers Inc., 1986.

[81] M. Kutsch, P. J. Haas, V. Markl, N. Megiddo, and T. M. Tran, "Integrating a maximum-entropy cardinality estimator into DB2 UDB," in *EDBT '06: Proceedings of the 10th International Conference on Extending Database Technology*, 2006.

[82] T. Legler, W. Lehner, and A. Ross, "Data mining with the SAP NetWeaver BI accelerator," in *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pp. 1059–1068, VLDB Endowment, 2006.

[83] H.-G. Li, S. Chen, J. Tatemura, D. Agrawal, K. S. Candan, and W.-P. Hsiung, "Safety guarantee of continuous join queries over punctuated data streams," in *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pp. 19–30, VLDB Endowment, 2006.

[84] W.-S. Li, V. S. Batra, V. Raman, W. Han, and I. Narang, "QoS-based data access and placement for federated systems," in *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pp. 1358–1362, VLDB Endowment, 2005.

[85] L. F. Mackert and G. M. Lohman, "R* optimizer validation and performance evaluation for distributed queries," in *VLDB '86: Proceedings of the 12th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 149–159, Morgan Kaufmann Publishers Inc., 1986.

[86] S. Madden, M. Shah, J. M. Hellerstein, and V. Raman, "Continuously adaptive continuous queries over streams," in *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 49–60, ACM Press, 2002.

[87] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh, and M. Cilimdzic, "Robust query processing through progressive optimization," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 659–670, ACM Press, 2004.

[88] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, "Query processing, resource management, and approximation in a data stream management system," in *CIDR '03: First Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, 2003.

[89] I. S. Mumick, S. J. Finkelstein, H. Pirahesh, and R. Ramakrishnan, "Magic is relevant," in *SIGMOD '90: Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 247–258, ACM Press, 1990.

[90] K. Munagala, S. Babu, R. Motwani, and J. Widom, "The pipelined set cover problem.," in *ICDT '05: Proceedings of the 10th International Conference, Edinburgh, UK*, pp. 83–98, 2005.

[91] K. Munagala, U. Srivastava, and J. Widom, "Optimization of continuous queries with shared expensive filters," in *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, (New York, NY, USA), pp. 215–224, ACM Press, 2007.

[92] J. Naughton, D. DeWitt, D. Maier, A. Aboulnaga, J. Chen, L. Galanis, J. Kang, R. Krishnamurthy, Q. Luo, N. Prakash, R. Ramamurthy, J. Shanmugasundaram, F. Tian, K. Tufte, S. Viglas, Y. Wang, C. Zhang, B. Jackson, A. Gupta, and R. Chen, "The niagara internet query system," *IEEE Data Engineering Bulletin*, June 2001.

[93] N. Polyzotis, "Selectivity-based partitioning: A divide-and-union paradigm for effective query optimization," in *CIKM '05: Proceedings of the 14th ACM International Conference on Information and knowledge management*, pp. 720–727, New York, NY: ACM Press, 2005.

[94] V. G. V. Prasad, *Parametric Query Optimization: A Geometric Approach*. Master's thesis, IIT Kanpur, 1999.

[95] V. Raman, A. Deshpande, and J. M. Hellerstein, "Using state modules for adaptive query processing.," in *ICDE '03: Proceedings of the 19th International Conference on Data Engineering, Bangalore, India*, pp. 353–364, 2003.

[96] V. Raman and J. M. Hellerstein, "Partial results for online query processing," in *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 275–286, ACM Press, 2002.

[97] V. Raman, B. Raman, and J. M. Hellerstein, "Online dynamic reordering for interactive data processing," in *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pp. 709–720, Edinburgh, Scotland: Morgan Kaufmann, 1999.

[98] S. V. U. M. Rao, *Parametric Query Optimization: A Non-Geometric Approach*. Master's thesis, IIT Kanpur, 1999.

[99] L. Raschid and S. Y. W. Su, "A parallel processing strategy for evaluating recursive queries," in *VLDB '86: Proceedings of the 12th International Con-

ference on Very Large Data Bases, pp. 412–419, Morgan Kaufmann Publishers Inc., 1986.

[100] P. Roy, S. Seshadri, S. Sudarshan, and S. Bhobe, "Efficient and extensible algorithms for multi query optimization," in *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 249–260, ACM Press, 2000.

[101] E. A. Rundensteiner, L. Ding, T. M. Sutherland, Y. Zhu, B. Pielech, and N. Mehta, "CAPE: Continuous query engine with heterogeneous-grained adaptivity," in *VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada*, pp. 1353–1356, 2004.

[102] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system," in *SIGMOD '79: Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, 1979.

[103] T. K. Sellis, "Multiple-query optimization," *ACM Trans. Database Syst.*, vol. 13, no. 1, pp. 23–52, 1988.

[104] P. Seshadri, J. M. Hellerstein, H. Pirahesh, T. Y. C. Leung, R. Ramakrishnan, D. Srivastava, P. J. Stuckey, and S. Sudarshan, "Cost-based optimization for magic: Algebra and implementation," in *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pp. 435–446, ACM Press, 1996.

[105] P. Seshadri, H. Pirahesh, and T. Y. C. Leung, "Complex query decorrelation," in *ICDE '96: Proceedings of the Twelfth International Conference on Data Engineering,* New Orleans, LA, pp. 450–458, February 26–March 1 1996.

[106] M. A. Shah, J. M. Hellerstein, and E. Brewer, "Highly available, fault-tolerant, parallel dataflows," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 827–838, ACM Press, 2004.

[107] J. Shanmugasundaram, K. Tufte, D. J. DeWitt, J. F. Naughton, and D. Maier, "Architecting a network query engine for producing partial results," in *ACM SIGMOD Workshop on the Web (WebDB) 2000,* Dallas, TX, pp. 17–22, 2000.

[108] M. A. Shayman and E. Fernandez-Gaucherand, "Risk-sensitive decision-theoretic diagnosis," *IEEE Transactions on Automatic Control*, vol. 46, pp. 1166–1171, 2001.

[109] H. Simon and J. Kadane, "Optimal problem-solving search: All-or-none solutions," *Artificial Intelligence*, vol. 6, pp. 235–247, 1975.

[110] U. Srivastava, K. Munagala, and J. Widom, "Operator placement for in-network stream query processing," in *PODS '05: Proceedings of the Twenty-Fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 250–258, 2005.

[111] U. Srivastava, K. Munagala, J. Widom, and R. Motwani, "Query optimization over web services," in *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pp. 355–366, VLDB Endowment, 2006.

[112] M. Stillger, G. Lohman, V. Markl, and M. Kandil, "LEO – DB2's LEarning Optimizer," in *VLDB '01: Proceedings of 27th International Conference on Very Large Data Bases*, Morgan Kaufmann, September 11–14 2001.

[113] M. Stonebraker, E. Wong, P. Kreps, and G. Held, "The design and implementation of Ingres," *ACM Transactions on Database Systems*, vol. 1, no. 3, pp. 189–222, 1976.

[114] M. Templeton, H. Henley, E. Maros, and D. J. V. Buer, "InterViso: Dealing with the complexity of federated database access," *The VLDB Journal*, vol. 4, no. 2, 1995.

[115] F. Tian and D. J. DeWitt, "Tuple routing strategies for distributed eddies," in *VLDB '03: Proceedings of 29th International Conference on Very Large Data Bases*, pp. 333–344, Berlin, Germany: Morgan Kaufmann, September 9–12 2003.

[116] P. A. Tucker and D. Maier, "Exploiting punctuation semantics in data streams," in *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, (Washington, DC, USA), p. 279, IEEE Computer Society, 2002.

[117] T. Urhan and M. J. Franklin, "XJoin: a reactively-scheduled pipelined join operator," *IEEE Data Engineering Bulletin*, vol. 23, no. 2, pp. 27–33, 2000.

[118] T. Urhan, M. J. Franklin, and L. Amsaleg, "Cost based query scrambling for initial delays," in *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pp. 130–141, Seattle, WA: ACM Press, June 2–4 1998.

[119] E. Viglas and S.-J. F. Naughton, *Novel Query Optimization and Evaluation Techniques*. PhD thesis, University of Wisconsin at Madison, 2003.

[120] S. Viglas, J. F. Naughton, and J. Burger, "Maximizing the output rate of multi-way join queries over streaming information sources," in *VLDB '03: Proceedings of the 29th International Conference on Very Large Data Bases*, Berlin, Germany: Morgan Kaufmann, September 9–12 2003.

[121] A. N. Wilschut and P. M. G. Apers, "Dataflow query execution in a parallel main-memory environment," in *PDIS '91: Proceedings of the First International Conference on Parallel and Distributed Information Systems,* Fontainebleu Hilton Resort, Miami Beach, FL, pp. 68–77, IEEE Computer Society, 1991.

[122] E. Wong and K. Youssefi, "Decomposition — strategy for query processing," *ACM Transactions on Database Systems*, vol. 1, no. 3, pp. 223–241, 1976.

[123] D. Zhang, J. Li, K. Kimeli, and W. Wang, "Sliding window based multi-join algorithms over distributed data streams," in *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)*, (Washington, DC, USA), p. 139, IEEE Computer Society, 2006.

[124] Y. Zhu, E. A. Rundensteiner, and G. T. Heineman, "Dynamic plan migration for continuous queries over data streams," in *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 431–442, ACM Press, 2004.