Query Optimization | DOI:10.1145/3347861

Turbocharging Database Query Processing and Testing

BY JAYANT R. HARITSA AND S. SUDARSHAN

DATABASE MANAGEMENT SYSTEMS (DBMS) constitute the backbone of today's information-rich society. A primary reason for the popularity of database systems is their support for *declarative* queries, typically in the SQL query language. In this programming paradigm, the user only specifies the end objectives, leaving it to the DBMS to automatically identify the optimal execution strategy to achieve these objectives. Declarative specification of queries is also central to parallel query execution in modern big data platforms.

Query processing and optimization have been extensively researched for close to five decades now, and are implemented in all contemporary database systems. Nevertheless, important challenges remain unsolved, and Indian universities have played a visible role in addressing these issues. As exemplars, we highlight recent research contributions on robust query processing, holistic optimization of database applications, and testing strategies for SQL queries and database engines.

Robust Query Processing

A crucial input to generating efficient query execution strategies, called *plans*, are the statistical estimates of the output data volumes for the algebraic predicates present in the query. In practice, these estimates, called *selectivities*, are often significantly in error with respect to the actual values subsequently encountered during query execution. The unfortunate outcome is a poor plan choice, resulting in query response times that may be worse by *orders of magnitude* relative to the optimal plan choice with the correct selectivities. A considerable body of literature exists on improving the statistical quality of selectivity estimates through sophisticated summary structures, feedback-based adjustments, and on-the-fly re-optimization of queries. However, a common limitation in this prior work is the inability to furnish performance *guarantees*.

A radically different approach that addresses the guarantee issue, called PlanBouquet,[a] has been recently developed at IISc Bangalore.[3] PlanBouquet completely *abandons* the classical estimation process for error-prone selectivities—instead, it employs a carefully calibrated "trial-and-error" sequence of time-budgeted plan executions that are progressively capable of handling more and more data until the query is eventually taken to completion. An advanced variant of this approach, called SpillBound, guarantees that the performance is *always* within a factor of (D^2+3D) relative to the ideal, where D is the number of predicates whose selectivity estimates may be erroneous.[5]

Further, empirical evaluations on industry-standard benchmarks have shown SpillBound to perform, in the worst-case, within a factor of 10–20 of the ideal, whereas contemporary database systems may suffer performance degradation factors running to the *1,000s* and beyond in such environments. This performance robustness of SpillBound is quantified in Figure 1 for a representative set of queries from the industry standard TPC-DS benchmark, the comparison yardstick being the PostgreSQL native optimizer.

These techniques represent an important milestone in the history of robust query processing since they are the first to provide quantitative performance guarantees, addressing a critical need of the database community.

Holistic Optimization

Database-backed applications often suffer from poor performance arising from sub-optimal ways in which imperatively written application programs access information from a database. For example, many application programs issue a long sequence of queries to a database, each of which requires a significant round-trip time due to latency in the database and network. Such inefficiencies cannot be addressed either by traditional database query optimizers or by traditional compiler optimizations. The DBridge system[b] developed at IIT Bombay therefore tackles this problem by rewriting application code to optimize data access.

DBridge carries out a set of potent equivalence-preserving transformations on imperative code

The DBridge suite of techniques brings the powerful benefits of declarative query optimization to imperative code.

a https://dsl.cds.iisc.ac.in/projects/QUEST

b https://www.cse.iitb.ac.in/infolab/dbridge

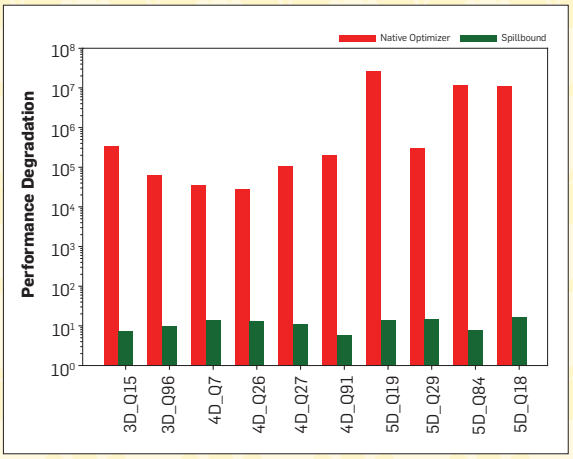

Figure 1. Performance robustness profile.

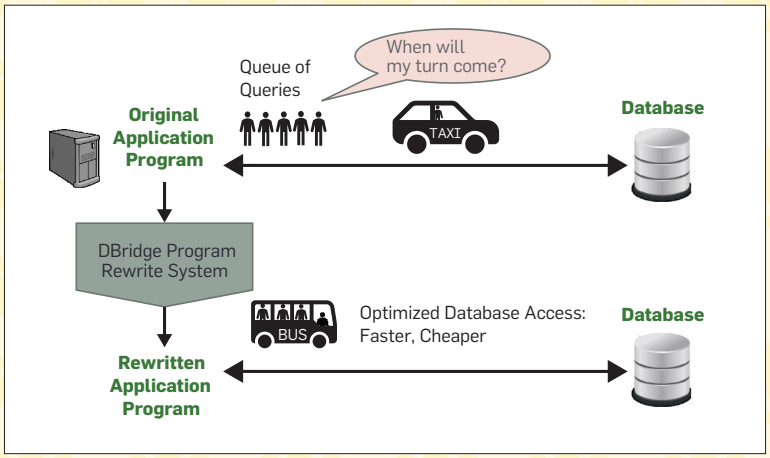

Figure 2. Rewrites for optimizing data access.

to speed up data access. The transformations successfully carry out batching and asynchronous submission of queries,[6] prefetching of query results, and conversion of procedural code to SQL. A metaphorical depiction of batching rewrites in DBridge is shown in Figure 2, where queries that are issued one-at-a-time, symbolized by the individual "taxis," are batched into a single unified request, carried by a "bus." Each transformation caters to a restricted scope and is therefore easy to prove correct, but in tandem they can successfully rewrite complex application programs. Further, the Cobra component of DBridge[4] efficiently chooses the least cost program from many alternative transformed programs, by leveraging concepts from query optimization based on algebraic equivalence rules.

Techniques for holistic optimization of queries containing imperatively coded user-defined functions (UDFs) were developed jointly by IIT Hyderabad and IIT Bombay; some of these mechanisms have subsequently been implemented and released in Microsoft SQL Server 2019, garnering excellent reviews from users.[c]

Collectively, the DBridge suite of techniques brings the powerful benefits of declarative query optimization to imperative code, opening a new research frontier. More details on these techniques may be found on the DBridge project home page.

Query and Engine Testing

With the onset of the Big Data world, where data is the engine driving virtually all aspects of human endeavor, it is vitally important to ensure both the applications and the underlying platforms are functionally correct. The XData system[d] developed at IIT Bombay supports testing of SQL queries by generating datasets designed to detect many types of common errors.[2] XData can be used in database courses to help students master the nuances of SQL query formulation and verify their correctness; further, the XData system facilitates *automated grading* of incorrect queries by assigning partial markings that reflect the severity of the errors. XData is currently operational at multiple universities.

The testing of Big Data platforms is addressed by the CODD project[e] at IISc Bangalore, using a distinctive metaphor of "dataless databases."[1] Here, databases with a desired set of characteristics can be efficiently *simulated* without explicit creation or persistent storage of the contents. This approach is essential since traditional testing techniques, which involve construction of representative databases and regression query suites, are completely impractical at Big Data scale due to the time and space overheads involved in their execution. The CODD tool has been successfully used for testing of database engines in the software and telecom industries.

Future Research

An important reason for the rapid adoption of SQL in the 1970s was its simplicity, which lent itself to effective query optimization. However, a host of complex features have been added over the years, and today's query processing world can be paraphrased as "with great expressive power comes great challenges." In this article, we have highlighted a few recent successes in tackling these challenges, but there remain rich opportunities for further contributions to the field. Productive future work areas include extending the holistic optimization concept to new domains (for example, machine learning), and leveraging query and data characteristics to deliver tighter robustness guarantees.

c https://www.microsoft.com/en-us/research/project/froid

d https://www.cse.iitb.ac.in/infolab/xdata

e https://www.cse.iitb.ac.in/infolab/xdata

References

1. Ashoke, S. and Haritsa, J. CODD: A dataless approach to big data testing. *PVLDB 8*, 12 (Aug. 2015), 2008–2011.
2. Chandra, B., Chawda, B., Kar, B., Reddy, K., Shah, S. and Sudarshan, S. Data generation for testing and grading SQL queries. *VLDB J. 24*, 6 (Dec. 2015), 731–755.
3. Dutt, A. and Haritsa, J. Plan bouquets: A fragrant approach to robust query processing. *ACM Trans. Database Syst. 41*, 2 (June 2016), 11–1:37.
4. Emani, K. V., and Sudarshan, S. Cobra: A framework for cost-based rewriting of database applications. In *Proceedings of the IEEE Intl. Conf. on Data Engg.* (Apr. 2018), 689–700.
5. Karthik, S., Haritsa, J., Kenkre, S., Pandit, V. and Krishnan, L. Platform-independent robust query processing. *IEEE Trans. Knowl. Data Eng. 31*, 1 (Jan. 2019), 17–31.
6. Ramachandra, K., Chavan, M., Guravannavar, R. and Sudarshan, S. Program transformations for asynchronous and batched query submission. *IEEE Trans. Knowl. Data Engg. 27*, 2 (Feb. 2015), 531–544.

Jayant R. Haritsa (haritsa@iisc.ac.in) is a professor at the Indian Institute of Science, Bangalore, India.

S. Sudarshan (sudarsha@cse.iitb.ac.in) is a professor at the Indian Institute of Technology, Bombay, India.