Query Optimization and Caching

Research Interests and Related Publications

Jarek Gryz
Department of Computer Science
York University, Toronto, Canada
jarek@cs.yorku.ca

• *Semantic Query Caching*

Query caching can play a vital role in heterogeneous, multi-database environments. Answers to a query that are available in cache at the local client can be returned to the user quickly, while the rest of the query is evaluated. The use of caches can optimize query evaluation. By caching certain sensitive data locally, caches can be used to answer the parts of queries that involve the sensitive data, so it need not be shipped across the network. Most prior cache schemes have been tuple-based or page-based. It is unclear, however, how these might be adapted for multi-databases. We explore a more flexible "semantic query caching" (SQC) approach. In SQC, a cache is the answer set of a previous query, labeled by the query expression.

We designed a logical framework for SQC in which caches are formally defined and by which one can determine when a query can be answered via cache. We specified the conditions to determine when answers, or partial answers, to a query are present in cache, and whether they can be retrieved from cache. These tests are complete in that all answers that can be retrieved through any combination of cache expressions will, in fact, be found. Based on this framework, we explore algorithmic solutions for the case when a query, possibly involving view predicates, is answerable entirely from cache. We are also interested in extending this framework to several new applications of SQC.

[1] P. Godfrey and J. Gryz, "Answering Queries by Semantic Caches" (in progress)

[2] P. Godfrey and J. Gryz, "Semantic Query Caching in Heterogeneous Databases", In *Proceedings of the 4th KRDB Workshop*, Athens, Greece, Aug. 1997, pages 6.1–6.6.

• *View Disassembly*

We explore a new form of view rewrite called ***view disassembly***. The objective is to rewrite views in order to "remove" certain sub-views (or ***unfoldings***) of the view. This becomes pertinent for complex views which may defined over other views and which may involve union. Such complex views arise necessarily in environments as data warehousing and mediation over heterogeneous databases. View disassembly can be used for view and query optimization, preserving data security, making use of cached queries and materialized views, and view maintenance.

We provide computational complexity results of view disassembly. We show that the optimal rewrites for disassembled views is at least NP-hard. We also provide an approximation algorithm that has much better run-time behavior. We show a pertinent class of unfoldings for which their removal always results in a simpler disassembled view than the view itself. We also show the complexity to determine when a collection of unfoldings ***cover*** the view definition.

[1] P. Godfrey, J. Gryz, "View Disassembly" (submitted)

[2] P. Godfrey, J. Gryz, "Overview of Dynamic Query Evaluation in Intensional Query Optimization", In *Proceedings of 5th DOOD*, Montreux, Switzerland, Dec. 1997

[3] P. Godfrey, J. Gryz, "A Framework for Intensional Query Optimization", In Proceedings of DDLP, Bonn, Germany, September 1996, pp. 57-68.

• *Query Containment via Resolution*

The problem of query containment received a lot of attention in the context of information integration. Several algorithms for testing containment for different types of queries have been considered and several complexity results established. We are exploring the issue of expressing the problem of query containment and query folding in logic programming, and using resolution to test for containment.

[1] J. Grant, J. Gryz, J. Minker, "Query Containment via Resolution" (in progress).

[2] J. Gryz, "Query Folding with Inclusion Dependencies", In *Proceedings of the 14th ICDE*, pp. 126-133, Orlando, FL, Feb. 1998.

[3] J. Gryz, "An Algorithm for Query Folding with Functional Dependencies", To appear in *Proc. of the 7th Int. Symposium on Intelligent Information Systems*, Malbork, Poland, 15-19 June, 1998.

• *Semantic Query Optimization*

Relational database systems became the predominant technology for storing, handling, and querying data only after a great improvement in the efficiency of query evaluation in such systems. The key factor in this improvement was the introduction and development of query optimization techniques. In the late 1970's and early 1980's, researchers recognized that the semantics of a database could be exploited for further query optimization, and developed a new set of techniques called semantic query optimization (SQO). SQO uses integrity constraints associated with the database to improve the efficiency of query evaluation. Although several different techniques for SQO have been developed, only simple prototypes have been built. To the best of our knowledge, no commercial implementations of SQO exist today. We are exploring ways to implement and test exisiting SQO techniques within the DB2 query optimizer developed by IBM. We are also working on extending exisiting SQO techniques to heterogeneous, multi-database environment.

[1] P. Godfrey, J. Grant, J. Gryz, and J. Minker, "Integrity Constraints: Semantics and Applications". In *Logics for Databases and Information Systems*, eds. G. Saake and J. Chomicki, 1998, pp. 265-307.

[2] J. Grant, J. Gryz, J. Minker, and L. Raschid. "Semantic Query Optimization in Object Databases" In *Proceedings of the 13th ICDE*, pages 444–453, Birmingham, UK, Apr. 7–11 1997.

[3] P. Godfrey, J. Gryz and J. Minker, "Semantic Query Optimization for Bottom-Up Evaluation". In *Proceedings of the 9th ISMIS*, Zakopane, Poland, June 1996, pp. 561-571.