Incremental evaluation of updates in factorized in-memory databases

Muhammad Idris, Martin Ugarte, Stijn Vansummeren
Universite libre de Bruxelles
Emails: {muhammad.idris, martin.ugarte, stijn.vansummeren} @ulb.ac.be

1 Abstract

Incremental view maintenance (IVM) is an established research area in the database and data-warehousing communities. In conventional IVM practices, query results are incrementally maintained to compute the delta of the query result, which is cheaper than full recomputation. A recent work in the DBToaster project [1, 4, 3] presents higher order IVM to maintain in-memory views under dynamic updates. It is based on the observation that when existing database D is modified with update u, the delta of the result $Q(D+u) - Q(D)$ can itself be expressed as a query ΔQ, whose results itself can be maintained. Updates to the result of ΔQ can be computed in a second order delta of Q, denoted $\Delta^2 Q$, which again can be maintained and so on untill a finite k^{th} order delta. Instead of evaluating Q on the new database $D+u$, we are interested in computing only the $\Delta^n Q$ for $1 \leq n \leq k$. However, materilizing each $\Delta^n Q$ quickly leads to a memory bottleneck, as we show.

In this research, we investigate so-called *factorized representations* of query results for continuous queries. Unlike relational databases, factorized representations, as presented by Oltenu et al. [2], are expressions where each expression item is either: a unary relation consisting of single data item, union of two expressions, or a product of two expressions. These are succinct representations where the product elements are distributed over unions. An example expression f of the join result of $Q = R(A,B) \bowtie S(B,C,D) \bowtie T(D,E)$ in Table 1 is given in the following.

$$f = a_1 \times b_1 \times c_1 \times d_1 \times (e_1 \cup e_2) \cup$$
$$a_2 \times b_2 \times (c_2 \times d_2 \times (e_1 \cup e_3) \cup c_3 \times d_4 \times e_5) \cup$$
$$a_3 \times (b_3 \times (c_3 \times d_3 \times e_3) \cup b_4 \times (c_3 \times d_3 \times e_4))$$

Table 1: Q=$R \bowtie S \bowtie T$

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_1	c_1	d_1	e_2
a_2	b_2	c_2	d_2	e_1
a_2	b_2	c_2	d_2	e_3
a_2	b_2	c_3	d_4	e_5
a_3	b_3	c_3	d_3	e_3
a_3	b_4	c_3	d_3	e_4

This representation exploits the dependency of attributes in a relation and the join of multiple relations can be represented as a join factorization tree where each relation is a hyperedge. While the existing works study factorizations for static data, we investigate incremental evaluation of updates on factorized representations. We employ a novel join tree decomposition method to reduce the memory footprints of a conjunctive query to linear in the size of input relations. Our algorithm is orders of magnitude more efficient in query evaluation time and memory footprints for full join queries, and approximately 2x times faster for aggregate queries.

References

[1] Yanif Ahmad and Christoph Koch. Dbtoaster: A sql compiler for high-performance delta processing in main-memory databases. *Proc. VLDB Endow.*, 2(2):1566–1569, August 2009.

[2] Nurzhan Bakibayev, Dan Olteanu, and Jakub Závodnỳ. Fdb: A query engine for factorised relational databases. *Proceedings of the VLDB Endowment*, 5(11):1232–1243, 2012.

[3] Christoph Koch. Incremental query evaluation in a ring of databases. In *Proceedings of the Twenty-ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, pages 87–98, New York, NY, USA, 2010. ACM.

[4] Christoph Koch, Yanif Ahmad, Oliver Kennedy, Milos Nikolic, Andres Nötzli, Daniel Lupei, and Amir Shaikhha. Dbtoaster: higher-order delta processing for dynamic, frequently fresh views. *VLDB J.*, 23(2):253–278, 2014.