Overview of Dynamic Query Evaluation in Intensional Query Optimization

Parke Godfrey[1], Jarek Gryz[2]

[1] U.S. Army Research Laboratory, Adelphi, Maryland, USA
[2] Department of Computer Science, York University, Toronto, Canada

In [1], We introduced a new query optimization framework called *intensional query optimization* (IQO). This framework enables existing optimization techniques to be applied to queries over views that employ unions. Advanced database technologies and applications such as federation and mediation over heterogeneous database sources, object oriented databases and query languages, and data warehousing for decision support readily lead to such complex view definitions, and thus tend to incur complex, expensive queries.

The IQO framework provides that some of the query's *expansions*, also called *unfoldings*, may be explicitly "separated out" of the query. It defines the notion of a *discounted query*, which is the query annotated to the effect that some of its expansions have been removed. (These removed expansions are called the *unfoldings-to-discount*.) Thus this offers us a way to recast complex queries, which use unions, into pieces for which we may have good optimizations.

The query's answer set is then equivalent to the union of the answer sets of the unfoldings-to-discount *plus* the answer set of the discounted query. For this to be a viable overall optimization strategy, it must be possible to evaluate a discounted query less expensively than the query itself; indeed less expensively than the query itself *less* the cost of evaluating each unfolding-to-discount.

We contend this is possible, and we motivate and sketch an efficient evaluation strategy for discounted queries. Consider the following example.

Let the database, **DB**, contain five base relations:

- **Faculty:** *Name* × *Department* × *Rank*
- **Staff:** *Name* × *Department* × *Years_of_Employment*
- **Ta:** *Name* × *Department*
- **Life_ins:** *Name* × *Insurer* × *Monthly_premium*
- **Health_plan:** *Name* × *Insurer* × *Monthly_premium*

Let there be two views in the **DB**: an *employee* relation, via the union of the *ta* relation and projections from the *faculty* and *staff* relations; and a *benefits* relation, via the union of projections from the *life_ins* and *health_plan* relations.

$$\begin{array}{ll} employee(X,Y) \leftarrow faculty(X,Y,Z). & benefits(X,Z) \leftarrow life\_ins(X,Z,W). \\ employee(X,Y) \leftarrow staff(X,Y,Z). & benefits(X,Z) \leftarrow health\_plan(X,Z,W). \\ employee(X,Y) \leftarrow ta(X,Y). & \end{array}$$

Define query Q to ask for the names of all employees of the physical plant, *p_p*, whose benefits are provided by *hmo*:

$$Q\colon \leftarrow employee(X,p\_p),\ benefits(X,hmo).$$

Assume that the following two queries $\mathcal{F}$ and $\mathcal{G}$ have been asked before, and that their results are stored in cache. Or, equivalently, assume that these formulas represent materialized views, or that they are guaranteed to evaluate empty by integrity constraints.

$$\mathcal{F}\colon \leftarrow ta(X,\ Y),\ life\_ins(X,\ hmo,\ \_).$$
$$\mathcal{G}\colon \leftarrow staff(X,\ p\_p),\ health\_plan(X,\ Z,\ \_).$$

It might be advantageous to evaluate the unfoldings of $\mathcal{Q}$ that correspond to these separately. This requires executing just select operations over $\mathcal{F}$ and $\mathcal{G}$ locally, to find a partial answer to $\mathcal{Q}$. Next, it is beneficial to "remove" these unfoldings from the query. The discounted query results in the rest of $\mathcal{Q}$'s answers.

In [3], we considered two strategies for the explicit removal of unfoldings-to-discount from a query. However, such *rewrite* strategies do not scale up for complex queries. Hence, we propose a dynamic evaluation approach that will scale up for queries over arbitrarily complex views. (See [2] for a complete description of the algorithm.) The strategy is a bottom-up materialization evaluation of the query tree, with the union and join operations modified to account for the unfoldings-to-discount. The strategy ensures two things:

1. that tuples from unfoldings-to-discount never contribute to the answer set of the discounted query; and
2. the join paths of the unfoldings-to-discount are never evaluated.

The idea is to keep extra information in the temporary tables created during the evaluation. In essence, each table has an extra column for each unfolding-to-discount, called a *tag* column. The value of a tag column for a given tuple indicates whether that tuple belongs to the answer set of its corresponding unfolding-to-discount. On each union or join operation, these tag columns' values must be maintained properly.

One can easily ensure the first property from above: after evaluation of the query, select only those tuples which have all *false* values in their tag columns (and project out the tag columns). One can further use the tag columns to ensure the second property, by determining which tuples should not be joined, because the resulting tuple would be from some unfolding-to-discount. The cost thus saved is the joining of the *true* sections of the two tables. If these sections (sub-tables) are large, this can be significant.

Thus IQO offers a viable approach towards optimizing complex, union queries.

References

1. P. Godfrey and J. Gryz. A framework for intensional query optimization. In D. Boulanger, U. Geske, F. Giannotti, and D. Seipel, editors, *Proceedings of the Workshop on Deductive Databases and Logic Programming at JICSLP'96*, pages 57–68, Bonn, Germany, September 1996.
2. P. Godfrey and J. Gryz. Intensional query optimization. Technical Report CS-TR-3702, UMIACS-TR-96-72, Dept. of Computer Science, University of Maryland, College Park, MD 20742, October 1996.
3. P. Godfrey, J. Gryz, and J. Minker. Semantic query optimization for bottom-up evaluation. In Z. Ras and M. Michalewicz, editors, *Proc. of the 9th Intl. Symp. on Meth. for Intell. Systems*, pages 561–571, Zakopane, Poland, June 1996.