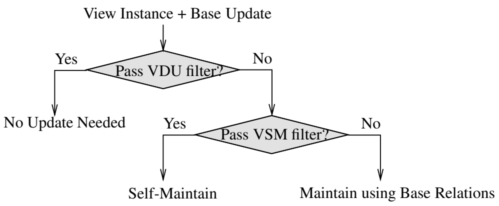
Speeding Up View Maintenance Using Cheap Filters at the Warehouse

Nam Huyn
Surromed, Inc., `phuyn@surromed.com`

Optimizing incremental view maintenance is an important problem in data warehousing. One popular technique, *view self-maintenance,* proposed computing the required view updates without using the underlying base relations. Traditionally, self-maintainability of a view is determined solely based on the view definition, independent of the state of the base relations. However, this "compile-time" approach to view self-maintenance is severely limited by the fact that most views are *not* self-maintainable. For these views, we propose *runtime view self-maintenance* which takes into account the current state of the views to decide view self-maintainability: given a view instance and a base update, we determine (1) whether or not the view is unaffected by the update (VDU), and if it is not, (2) whether or not the view is self-maintainable (VSM). Our plan for speeding up view maintenance can be depicted as follows:



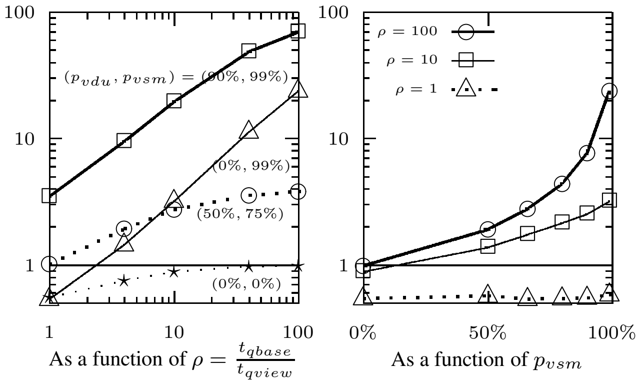
where the VDU and VSM tests are evaluated against the view instance at runtime. Obviously, if a view is compile-time self-maintainable, we would not have to do the tests at runtime. The strength of our approach lies in its ability to handle the more difficult case where a view is not compile-time self-maintainable. Unfortunately, even for CQ views (aka Select-Project-Join), VDU and VSM are hard decision problems (co-NP-complete in the size of the view instances). The challenge is to identify cases that can be efficiently tested, assuming that we have efficient ways to implement only table lookups (e.g., by using indexes.)

In this paper, we identify two important classes of CQ views that allow VDU and VSM to be tested efficiently: (a) views with *no self-joins* and (b) views with only *exposed self-joins*. The following table summarizes the number of view lookups needed to implement our tests:

	Insertion	**Deletion**
Testing VDU	(a) 1. (b) $(k-2)^2+1$.	1
Testing VSM	(a) $k-1$. (b) $(k-1)^2$.	1
Self-Maintenance	$k-1$	1

where k is the number of subgoals in the view definition (we refer the reader to [1] for the algorithm details.) Our solution may seem restrictive, since it does not handle self-joins in general. However, let us emphasize that our solution is both necessary and sufficient. For CQ views using arbitrary joins, we show how to reduce the problem to a problem with only exposed self-joins which we can solve efficiently. While we may loose completeness, our view maintenance plan remains sound.

How much speed-up can we achieve? The answer depends on how expensive it is to query the base relations and how often our tests succeed, allowing shorcuts to be taken. To study the performance of our approach, we implemented a view manager in Oracle PL-SQL which maintains a simple view under various synthetic update streams and under various delays in querying the base relations. The following graphs show the speedup factor as a function of ρ, the relative cost of querying the base relations, and as a function of p_{vsm}, the frequency of self-maintainability, when the view is always affected by the base update (i.e., when $p_{vdu} = 0$.)



The measured speedups agree well with the theoretical analysis we show in the paper. While further performance studies remain to be done, we believe our technique can significantly speed up incremental data warehouse maintenance.

References

[1] N. Huyn. *Speeding Up Materialized-View Maintenance Using Cheap Filters at the Warehouse*. Full manuscript, available at `www-db.stanford.edu/pub/papers/vmf.ps`.