

An agent-based Framework for materialized view maintenance in Collaborative Electronic Commerce Environments

Aristides Triantafillakis

Department of Informatics and Telecommunications

UNIVERSITY OF ATHENS

University Campus, Athens

Phone: +302107275217

Greece, 15771

Triant@di.uoa.gr

Panagiotis Kanellis

Department of Informatics and Telecommunications

UNIVERSITY OF ATHENS

University Campus, Athens

Phone: +302107275217

Greece, 15771

Kanellis@di.uoa.gr

Drakoulis Martakos

Department of Informatics and Telecommunications

UNIVERSITY OF ATHENS

University Campus, Athens

Phone: +302107275217

Greece, 15771

martakos@di.uoa.gr

Abstract

For extended enterprises, value-chain integration means that enterprise business systems can no longer be confined to internal processes, applications, and data repositories. What this means for data warehouses, is that their interoperation is a prerequisite in collaborative environments such as Business-to-Business exchanges where they are physically scattered along the value chain. Adopting system and information quality as success variables, we argue that existing works fell short of addressing complex issues that relate to their refreshment and extent far beyond view maintenance solutions within single warehouses. Describing a solution for warehouse refreshment in a federation of data warehouses, we define a special kind of materialized view and provide a prototype for materializing this kind of views.

Keywords

Decision Support Systems, Multi-agent Systems, Data Warehouse Refreshment, Materialized View Maintenance.

1. Introduction

A base requirement for the success of a data warehouse (DW) is the ability to provide decision makers with both accurate and timely consolidated information (information quality) as well as fast query response times (system quality) (Chen et al., 2000). For this purpose, a common method that is used in practice for providing higher information and system quality is the concept of materialized views where a query is more quickly answered against the materialized view than direct querying the base data sources. However, materialized views can improve query performance only if we can manage to update them consistently (Do et al., 1998).

Collaborative electronic commerce (Ce-commerce) augments this challenge because the data sources are not only internal, as they largely were a mere few years ago. For example, the emergence of business communities in the form of Business-to-Business (B2B) exchanges means that the boundaries of organizations are more fluid than they used to be (Yang and Papazoglou 2000). 'Extended enterprises' have to integrate far more data into a single repository originating outside the organization.

What this means for data warehouse information and system quality is that we should start considering the Data Warehouse Refreshment (DWR) as a process that must provide explicit support for cross-enterprise collaboration. Therefore, DWR should not only be limited to materialized view maintenance in the case of a single warehouse (as we are accustomed to) but also support the refreshment in a federation of data warehouses. This, in turn, translates to providing a new set of algorithms and techniques to materialize views from source data that may reside in materialized views of remote data warehouses, and to incrementally maintain these views. To the best of our knowledge, previous work on view maintenance has mainly considered the case of SPJ (Select-Project-Join) views in a single warehouse, not providing insights for a data warehousing architecture in such an environment. In short, the problem addressed in this paper is as follows: "how to maintain materialized views in environments where a data warehouse utilize data from other

Adopting system and information quality as success variables (DeLone and McLean 1992) we argue that DWR is more complex than the materialized view maintenance problem and we introduce a special kind of materialized views that emanates from such an environment.

In the next section we demonstrate the DWR process in this environment, in Section 3 we give a high-level view of the prototype's architecture and we present initial findings from the evaluation of a prototype that was build to test our approach in section 4. Our conclusions follow in section 5.

2. Data Warehouse Refreshment in Collaborative E-Commerce

An integrated value system, that is the case of a federation of DWs, necessitates a solution where the enterprise system of each participating business in the value chain should be able to communicate with other such systems in order to accomplish a task. In particular, a form of B2B exchange that has gained a lot of attention due to its potential benefits is the Continuous Replacement Planning (CRP), where a manufacturer supplies the distributors before requested. In this way multiple enterprises can collaboratively plan and manage the flow of goods, services, and information along the value chain in a way that increases customer-perceived value and optimises at the same time internal efficiencies (Yang et al., 2000).

For CRP, the root of the value chain should be able to communicate with the other participating enterprises, or in other words, to able to query the distributor's data repository in order to supply

before requested. However, considering system and information quality as success variables a decision maker in the root of the value chain (e.g. production manager) should demand fast query response times (i.e. information quality) to business questions and not be constrained by corporate boundaries (i.e. system quality). That is, he should query aggregated result sets for each product and distributor it procures and not detailed data, in order to better plan for the production lines. In this way, the root of the value chain should be able to query the distributor's DW in order to analyse aggregated result sets. This in turn is translated in customizing and querying remote materialized views in each participating business that contains information of interest to the collaborative-business workflow. Thus, the DWR process should also refer to defining materialized views that depend on remote materialized views of each participating DW.

Therefore, we define a materialized view in this context as a hyper-view (HV), because it provides a higher level of aggregation and consolidation of data sources that may reside in remote DWs of the participating enterprises in the value chain. The main characteristics of a HV that distinguish it from a normal materialized view are as follows:

- 1) The data sources may be materialized views themselves. Figure 1 depicts this situation, where for simplicity we have considered two enterprise systems, namely Site1 and Site2. Hyper-views pump data from the materialized views of each site and not necessarily from base tables, as we are accustomed to.
- 2) A union-based HV can be stored in a central DW that is remote from the participating DWs. Hyper-views belong to the customisation step of the DWR process and may have various forms. For example, they may be defined as a union over the corresponding materialized views of the participating warehouses, or as an aggregate query that provides decision makers with a higher level of aggregated and consolidated information.

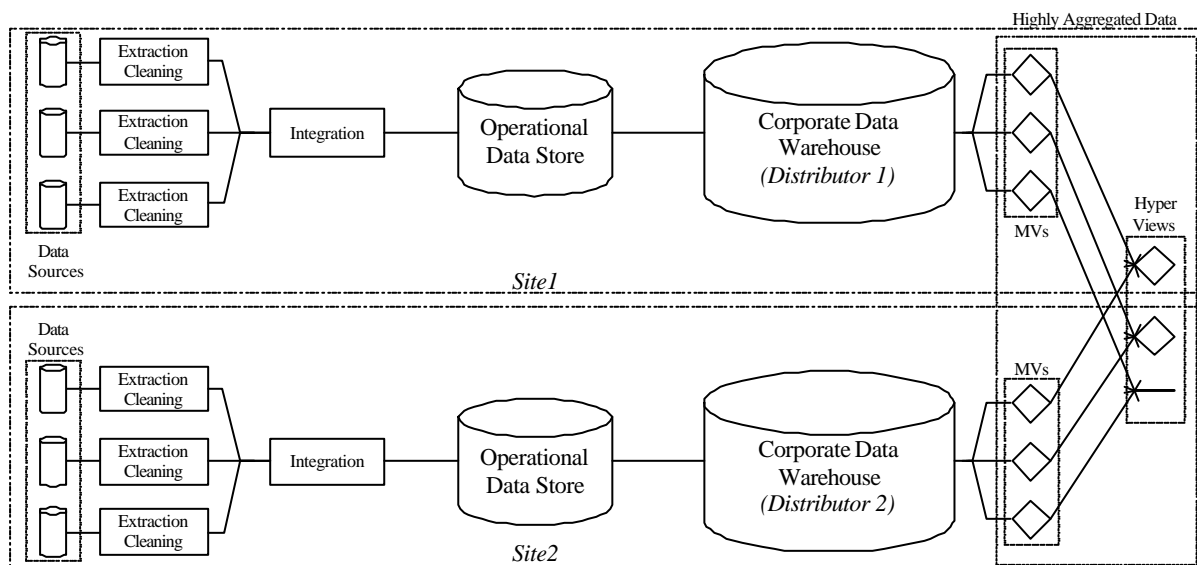


Figure 1. Data warehouse architecture in a Ce-Commerce environment

As we will see next, considering a HV as a union over remote materialized views is not a trivial issue and is only partially addressed in practice, by creating a materialized view in each database that does the aggregation for that database and then combine the results. It is obvious that the result set is not stored in a central repository, so the union-based view is not materialized but recomputed on-demand. Consequently, a HV that stores union-based result sets has greater performance, which in turn translates to better system quality.

Hence, from the above discussion we can say that the DWR process is a complex and event-driven system that needs certain monitoring and evolves frequently, following the evolution of data sources and user requirements. Thus, it refers to the ability to define different scenarios depending on user requirements, source constraints and data warehouse constraints. However, most of the prior work on warehouse refreshment deals with the problem of maintaining SPJ-type materialized views incrementally in the case of a single warehouse. Some algorithms presented in the literature (Zhuge et al., 1995), (Quass et al., 1996), (Zhuge et al., 1996), (Agrawal et al., 1997), (Zhuge et al., 1997), (Wang et al., 1999) should adopt a different policy that takes into account specific constraints such as freshness of data, space limitation of the ODS (Operational Data Store) or DW, referential integrity constraints on source data and access frequency to sources that users, data warehouse administrators and data source administrators may impose respectively. In particular, the Strobe Algorithm (Zhuge et al., 1996) has the potential of infinite waits, that is, it requires quiescence from the sources to update the corresponding views. It also ensures strong consistency but not complete consistency since it incorporates the effects of several updates collectively. In the case of the ECA algorithm (Zhuge et al., 1995) the data warehouse model is restricted in that the number of data sources is limited to a single data source. It is obvious that in a Ce-commerce environment, quiescence is unlikely and transactions involve multiple entities. Although, the SWEEP algorithm (Agrawal et al., 1997) guarantees complete consistency and has linear message complexity with the number of data sources for processing an update, it does not consider access frequency constraints to the data sources. Again, approach described in (Wang et al., 1999) is restricted to a single warehouse. Although the authors in (Quass et al., 1996) use key and referential integrity constraints in order to minimize auxiliary data and make a class of views self-maintainable, the class of views considered does not include aggregation. An architecture for multi-view consistency is proposed in (Zhuge et al., 1997), using an integrator, view managers and a merge process that collects changes to the views, holds them until all affected views can be modified together, and then forwards all of the views' changes to the DW in a single warehouse transaction. However, the authors provide algorithms for the merge process to decide when to hold and when to forward actions.

Regarding commercial database products, the problem described in this paper has not been adequately addressed either. For example, a materialized view in Microsoft SQL Server 2000 (called indexed view) must not reference any other views, only base tables and all base tables referenced by the view must be in the same database as the view. Besides, UNION is not allowed in an indexed view. A workaround to these issues is to create an indexed view in each database that does the aggregation for that database and then combine the results. Thus, the combined result set is not materialized, so we lose one of the fundamental properties of a DW that of stored result sets.

In short, the case of a federation of DWs that are physically scattered along the value chain in collaborative environments on the web such as Business-to-Business exchanges, necessitates an architecture where interoperability is a prerequisite. In this environment highly aggregated materialized views should depend on remote materialized views of the participating enterprises. Prior work on DWR has mainly considered the incremental maintenance of SPJ-type materialized views in the case of a single warehouse and in practice UNION based views cannot be materialized.

Consequently, we believe that a different approach is needed that deals with the incremental materialized view maintenance problem in the defined Ce-commerce environment. We define the data warehouse architecture in this environment as a hyper-warehouse (HW), because it is a superset of the participating data warehouses. Thus, we consider a hyper-view as a union over its corresponding materialized views (MV1, MV2, ..., MVn) of the n participating warehouses. We

also utilize agents (Zhuge et al., 1997), (Ding et al., 1999) for the interoperation of the DWs. This is because agent software can be installed on top of an RDBMS in every participating DW, that transmits the log of changes (base table deltas) to an integrator entity at the HW, which is responsible for maintaining the HV. The child-agent entity pumps data directly from the existing materialized view. We modelled this by installing triggers on the materialized view that replicate the changes to a temporary table (delta table - ΔR) at a participating DW. Later, an algorithm based on agent technology incrementally maintains the HV at the HW by analysing the base table deltas at the ΔR and posting the appropriate queries to the HV, where updates are handled as deletions followed by insertions.

3. Implementation Issues

One issue that arose during the design of a prototype was the method of populating the base table deltas. There are two options. One is to define triggers on each base table R , so that updates, insertions, and deletions will trigger the insertion of change records into ΔR . The other option is to populate ΔR by extracting changes from the database engine's transaction log. However, we choose the trigger method because it does not require knowledge of the database engine's log format and is simpler to implement.

Another issue was the fact that triggers on indexed views are not fired as part of the RDBMS's view maintenance internal process. Thus, we modelled the materialized view of interest of each DW as a table and installed triggers on the referenced base tables that maintain the materialized view (considering it as a table) incrementally. However, it would be far better if there was a special kind of triggers that could be fired as part of the RDBMS's view maintenance internal process. In this case we would not model the materialized view as a table and we could apply our algorithm directly to the materialized view.

4. Experimental Analysis

In a previous paper (Triantafillakis et al., 2002) we give the complete algorithms for the child-agent and integrator entities. A prototype of the proposed algorithms has been implemented using a set of external drivers around the Microsoft SQL Server 2000 relational database engine. Figure 2 gives a high-level view of the prototype's architecture. Solid arrow-lines represent data flow.

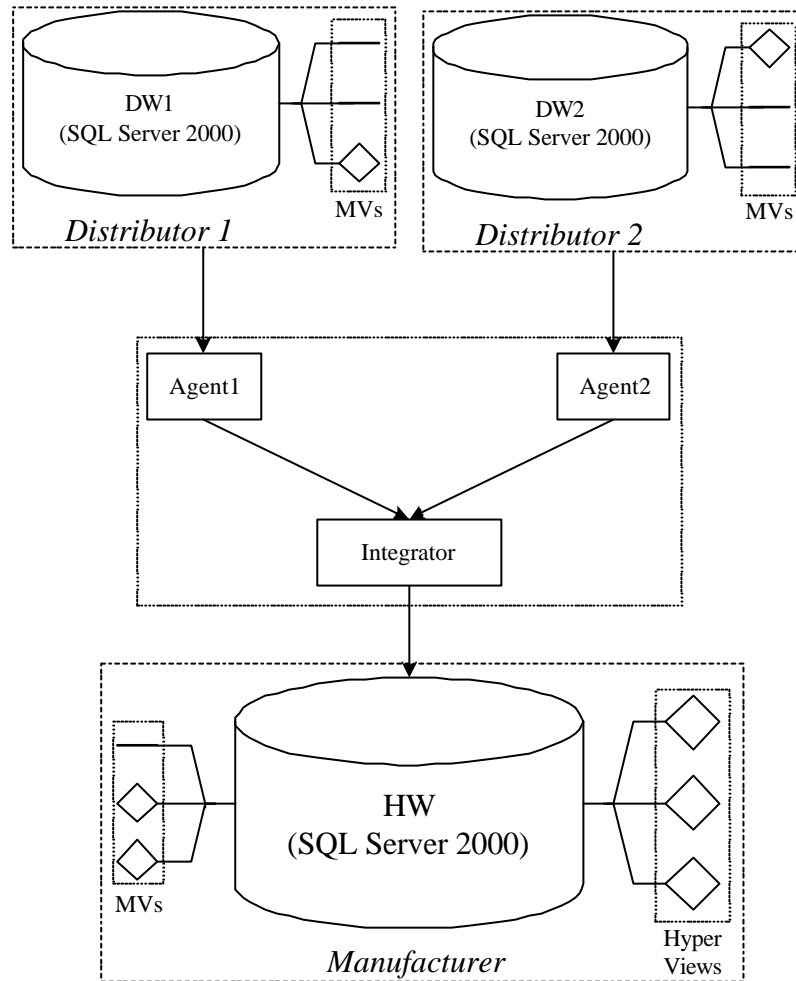


Figure 2. High-level view of the prototype's architecture

Performance is evaluated on two aspects: answering queries over union-based non-materialized hyper-views that depend on materialized views and relating the cost of performing a query over a non-materialized view to the cost of propagating incremental updates to the corresponding HV. In accordance with the evaluated aspects above, the hyper-views used are referred to as C_HV1 and C_HV3.

In Table 1 we illustrate the evaluation queries used that a decision maker at the HW would be interested in having fast system response time.

Query Name	Query Definition
Qa	Select * from c_hvx where availqty <= 40 order by p_name
Qb	Select p_name, sum(availqty) from c_hvx group by p_name having sum(availqty) <= 40

Table 1. Queries used for the evaluation process

Query Qa selects all the products from the participating materialized views that the manufacturer produces where the available quantity of each product is less than or equal to 40 pieces. Query Qb

selects the product name and the sum of the available quantity of each distinct product in each participating DW, where the sum is less than or equal to 40 pieces.

In the first case, C_HV1 is defined as a union query that selects the product name and the available quantity of each product and participating remote materialized view. This is accomplished through the linked server and 'openquery' properties of Microsoft's SQL Server 2000. Specifically, the definition of the C_HV1 is as follows:

```
Create view C_HV1 as
(select p_name, availqty, 1 as mySUPPLIER from openquery(DW01,'select p_name,
availqty from C_MV ')
union all
select p_name, availqty, 2 as mySUPPLIER from openquery(DW02,'select p_name, availqty
from C_MV'))
```

Figure 3. The HV used for answering queries over union-based materialized views.

As we will see next, the performance of this union query has almost constant response time with the updates that take place at the participating DWs. Although the result set is always up-to-date with the source data, this case has the slowest response time of the presented cases because the result set is not materialized at the HW but recomputed on demand. The only materialized views are those at the participating DWs. In addition, this case is susceptible to network/link failures because the HW must query remote DWs.

The hyper-view (C_HV3) is defined as a table at the HW and we populate the C_HV3 by issuing a query similar to that of Figure 3. Although this case has an initial materialization overhead, consequent updates to the participating DWs are applied incrementally to the C_HV3. This is accomplished through the use of the agent and integrator entities where the former transmits the base table deltas and the later installs the updates to the HV by issuing the appropriate queries (Triantafillakis et al. 2002). In this way, the hyper-view is always up-to-date with the source-materialized views and consequent queries are run against this table only (i.e. C_HV3) providing enhanced system and information quality, that is, fast system response time and up-to-date data. However, this case is susceptible to network/link failures because the agent and integrator properties communicate using TCP/IP.

We ran the evaluation queries (i.e. Qa and Qb) on every C_HVx and performed updates at the source-materialized views on different times, namely t1 and t2, where t0 is the initialisation phase. Table 2 compares the average system response time of the evaluating queries that retrieve the result of Qa and Qb, that is, the available quantity of each product and participating distributor, where it is less than or equal to 40 pieces.

HV	t1		t2 (Update)		t3		t4 (Update)		t5	
	Qa	Qb	Qa	Qb	Qa	Qb	Qa	Qb	Qa	Qb
C_HV1	20,2	37,2	20,5	37,1	20,2	37,2	20,2	37,2	20,2	37,2
C_HV3	28,6	30	1	2,8	0,51	2,4	1	2,8	0,51	2,4

Table 2. Average system response time of the evaluation queries

As we can see, the case of answering queries over union-based non-materialized hyper-views that depend on materialized views (i.e. C_HV1) has almost constant system response time. This was

expected due to the fact that the C_HV1 is not materialized but recomputed on demand. Moreover, the proposed architecture (i.e. case of C_HV3) outperforms the other approach, due to the fact that updates are applied incrementally to the HV. Thus, consequent queries are evaluated only against the HV, which is a local table/view at the HW.

Further Research

In this paper we have presented an original approach to view materialization in which a materialized view depends on remote materialized views and identified this materialized view as a hyper-view. We also described a real world scenario where the concept of hyper-views is most applicable, that is the case of the Continuous Replacement Policy.

However, we modelled a materialized view as a table and provided a simple maintenance technique by installing triggers on source tables. This is due to the fact that a trigger on an indexed view is fired upon explicit insert/update/delete on the view itself. We believe it would be more convenient and less computationally expensive if there was a special kind of triggers on an indexed view that were fired upon the RDBMS's view maintenance internal process. Moreover, in our description we assumed that the base tables in each data warehouse are available and we installed triggers on them. In this way, we bypassed an intermediate level that of a materialized view. Further research should concentrate on installing triggers directly on the materialized views instead of the base table.

Our continuing research efforts concentrate towards conducting experimental analysis of the proposed architecture in comparison with the existing approach offered in commercial database systems. For this purpose we will use the TPC-D 1GB database for benchmarking in Microsoft SQL Server 2000. There will be two clients (DWs) and one Server (HW) connected in a LAN. The testing will measure the performance of querying aggregated and consolidated result sets from the two DWs using some TPC queries. We strongly believe that our architecture will have better results with respect to the existing approach because the HV is materialized and not re-computed on demand.

Concluding, business value chains are increasingly transformed into integrated value systems, creating and enhancing customer-perceived value by means of cross-enterprise collaboration. For such enterprise landscapes current research on DWR offers little or no insight as it focuses mainly on update propagation through materialized views for single warehouses. We believe that this paper underlines a new set of problems that demand our attention, far beyond view maintenance, which is just one step of the complete refreshment process. Other steps may concern, data cleansing and data merging due to potential data and/or schema differences between the participating warehouses, and data customisation through the concept of the hyper-views.

References

- Agrawal, D, Abbadi, A, Singh A & Yurek, T (1997), *Efficient View Maintenance at Data Warehouses*. In The Proceedings of the ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA, pp. 417-427.
- Chen, LD, Soliman, K, Mao, E & Frolick, N (2000), *Measuring User Satisfaction with Data Warehouses: an Exploratory Study*. Information and Management, vol. 37, pp. 103-110.
- DeLone, W.H. & McLean, E. R. (1992), *Information Systems Success: the Quest for the Dependent Variable*. Information Systems Research, vol. 3, no. 1, pp. 60-95.

- Ding, L, Zhang, X & Rundensteiner, E.A (1999), *The MRE Wrapper Approach: Enabling Incremental View Maintenance of Data Warehouses Defined on Multi-Relation Information Sources*. In the Proceedings of the ACM second international workshop on Data warehousing and OLAP, Kansas City, Missouri, USA, pp. 30-35.
- Do, L, Drew, P, Jin, W, Jumani, V & Rossum, D.V (1998), *Issues in Developing Very Large Data Warehouses*. In the Proceedings of the 24th International Conference on Very Large Databases, New York City, New York, USA, pp. 633-636.
- Quass, D., Gupta, A & Mumick, I.S (1996), *Making Views Self-Maintainable for Data Warehousing*. In the Proceedings of the 4th International Conference on Parallel and Distributed Information Systems, Miami Beach, Florida, USA, pp. 158-169.
- Triantafillakis, A., Kanellis, P & Martakos, D (2002), *Data Warehouse Clustering on the Web*. In the Proceedings of the 13th International Workshop on Database and Expert Systems Applications, Aix-en-Provence, France, pp.800-804.
- Wang, H, Orłowska, M & Liang, W (1999), *Efficient Refreshment of Materialized Views With Multiple Sources*. In the Proceedings of the International Conference on Information and Knowledge Management, Kansas City, Missouri, USA, pp. 375-382.
- Yang, J & Papazoglou, M (2000), *Interoperation Support for Electronic Business*. Communications of the ACM, vol. 43, no. 6, pp. 39-47.
- Zhuge, Y, Garcia-Molina, H, Hammer, J & Widom, J (1995), *View Maintenance in a Warehousing Environment*. In The Proceedings of the ACM SIGMOD International Conference on Management of Data, San Jose, California, pp. 316-327.
- Zhuge, Y, Garcia-Molina, H & Wiener, J.L (1996), *The Strobe Algorithms for Multi-Source Warehouse Consistency*. In the Proceedings of the Conference on Parallel and Distributed Information Systems, Miami Beach, Florida, USA, pp. 146-157.
- Zhuge, Y, Wiener, J.L & Garcia-Molina, H (1997), *Multiple View Consistency for Data Warehousing*. In the Proceedings of 13th International Conference on Data Engineering, Birmingham, U.K, pp. 289-300.