Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1997 Proceedings

Americas Conference on Information Systems (AMCIS)

8-15-1997

A Data warehouse within a Federated database architecture

Anand V.J
National University of Singapore, sundar1@ix.netcom.com

A Data warehouse within a Federated database architecture

Anand, V. J.
Dept. of Decision Sciences, National University of Singapore
Email: sundar1@ix.netcom.com

Introduction

Research in heterogeneous databases [Sheth & Larson 90] have provided methods to integrate disparate databases into a single unifying architecture - the federated database model. But they are limited in as much as: 1) The federated schema is non-materialized, which means that queries will have to be evaluated in the individual databases, resulting in slower response time, and 2) Data from external sources are not integrated within the federated schema. We propose to extend the federated architecture to include a data warehouse [Inmon 94, Kimball 96] modeled as a materialized view [Hanson 87] of the underlying federated schema. In addition, we employ view maintenance techniques to maintain the data warehouse against changes in the underlying operational sources. We adopt a *deferred* view maintenance [Colby et al 96] approach, rather than *immediate* approach adopted by Stanford WHIPS project [Hammer et al 95]. This approach is preferred, because a great deal of decision-making may not require current data, but for those that require them, the model provides a mechanism to obtain them without adding too much overhead. For example, a data warehouse at a central office of a large chain of stores would like to have access to current inventory levels at individual stores, before deciding on a promotion. Similarly, access to both historical and current data of stock prices help an investment company to re-create point-in-time snapshots to help predict movements in stock prices.

This approach provides the following advantages:

- A unified architecture that ties the data warehouse to multiple heterogeneous databases.
- Provides a method of maintaining the data warehouse as an integrated materialized view of the underlying data sources.
- Provides flexible access to current data residing in the data sources.
- Ease of maintenance against any change to the schema in the data source or warehouse.

The Integrated Data Warehouse Model (IDWM)

IDWM integrates a data warehouse into a tightly coupled federated database architecture (Figure 1). This provides a three tier architecture: operational data sources and data warehouse forms the bottom and top respectively, while the interface layer provides the access methods between the two. When access to the current data is required, it is retrieved transparently from the operational data sources by accessing the interface layer.

In the model, a *component database* (CDB) refers to an operational data source. Data that is of interest is first represented uniformly through the *export schema* - filters and combines data defined in the component database into meaningful views. After reconciling, integrating and removing semantic heterogeneity between various export schemes, these are then integrated along with the external data into an *integrated data schema* (IDS) depicted near the top of Figure 1.

Here, the distinction between IDS and the data warehouse schema is that the former provides a consistent and reconciled view of all the data existing in the enterprise including external data, while the later is a subset of IDS and tailored to the personalized needs of a user. The interface layer provides accessing methods between the component databases and the data warehouse. This layer consists of software modules that implements utilities to generate differential files, mapping routines between the various schemes, communication sub-systems and message handlers.

IDWM does not assume on any specific data model adopted by the underlying data sources although data warehouse is assumed to be a non-legacy database. But for the purposes of illustration and developing mechanisms, we consider component databases and the data warehouse to be relational providing support for trigger mechanism, and view definitions. We emphasize that these assumptions do not in any manner make the model less generic.

IDWM maintains the warehouse as a materialized view over the component databases.

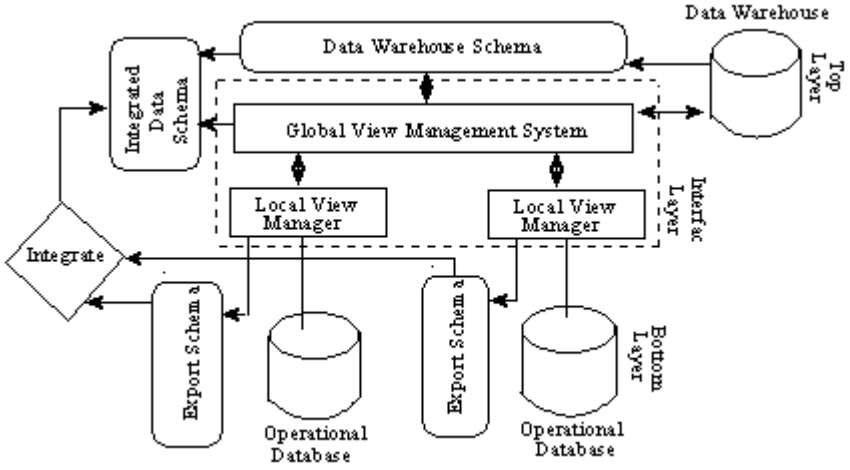

Figure 1: The Integrated Data Warehouse Model

In order to maintain the views the model uses the deferred method for propagating incremental changes. This method allows the materialized view and base tables to be out-of-sync for a period of time, which is tolerable in the warehousing scenario. But additional efficiency is gained by batching several updates together, and keeping the transaction and communication overhead to the minimum. The method used in this model consists of applying a set of propagation rules to a materialized view defined as select, project, join or any combination thereof. The rules [Colby et al 96] compute the tuples

that need to be deleted and inserted into the materialized view by accessing the base tables and its transaction logs alone, without referencing the materialized views in the data warehouse.

Global View Management System (GVMS), and the *Local View Manager* (LVM) implement the view management functions in IDWM. LVM provides the interface between operational database and data warehouse through the GVMS. Also, it provides utilities to generate differential files and specific mapping utility between the export and local schema. In addition, it periodically sends update from the operational source to the warehouse. GVMS provides similar functionalities to the data warehouse as LVM to the operational databases. In addition, it services request for data from the warehouse and coordinates LVMs during the periodic updates. During update propagation GVMS first propagates the change to the materialized export views, before propagating to the warehouse views.

Local View Manager

This is a software module that provides the interface between a component database and the data warehouse. Generally, there may not be a direct mapping between the local and export schemes, e.g., a table in the export schema can be a view defined by a join between tables. Therefore, a comprehensive catalog is maintained which provides mapping information. Moreover, it makes it easier for the local administrator to maintain the catalog for any change in the local schema. The other major functionality of LVM is to generate differential files for the views defined in the export schemes. The implementation of this function depends on the individual component database. For a database that support views and trigger mechanism LVM generates a transaction file for each table. At the time of propagation a differential file is generated for every view.

Global View Management System

GVMS is a software module that provides similar functionalities to the data warehouse as LVM provides to the component databases. But the GVMS is a global master that coordinates the LVMs and determines the next refresh cycle.

Functionally, GVMS provides:

- A utility to map IDS to the warehouse schema, and also IDS to the export schema of component databases.
- Algorithm to generate differential files for updating the data warehouse schema.
- Coordination of propagation of differential files from LVMs.
- Requests to LVMs to propagate differential files for ad-hoc data.

GVMS consists of various software modules that perform different functions. The *Mapper* transforms the warehouse schema to the underlying export views by interacting with IDS. The *Assembler* consolidates results and propagates updates to materialized views, and it also implements the view maintenance algorithm in GVMS. Coordination

between these modules is performed by the *Coordinator*, while *Monitor* provides interface between GVMS and LVMs. For ad-hoc queries the coordinator passes the definition to the mapper in order to find relevant export views that fulfill the request. Next, the mapper passes the site addresses of the relevant export views to the monitor to retrieve the differential files. These are buffered until all requests are obtained by the monitor, and passed on to the assembler which then assembles them according to the query definition, and submits the result to the coordinator. For periodic update propagation from the component databases the monitor has an elaborate tracking mechanism, through which it ensures that all the differential files are received by the warehouse, before the assembler begins propagating. In the event of a network failure the monitor requests the corresponding local view manager to re-send the differential files.

Conclusions

A great deal of work has been carried out in heterogeneous databases, view maintenance and data warehousing, but only recently have there been some research in integrating them. Our aim is to develop a model and define interfaces required to integrate these methods into a unified architecture. In this process many open and challenging issues that require further investigations were brought to focus, such as: (1) View maintenance algorithms for updating aggregates - there are not many generalized algorithms to handle these operations, (2) We assume warehouse as an append only database, but what about those that can have updates as well, e.g., to enable error-correction and schema changes, (3) Expiring data from the warehouse. We are presently working on extending the various functional modules that have been defined in the IDWM to include non-relational databases. Further we are working on defining a high level operator that performs drill-down from consolidated warehouse data to include most recent data from the component databases.

Reference

1. Colby, S. L., T. Griffin, L. Libkin, I. S. Mumick, H. Trickey, Algorithms for Deferred Maintenance, *Proceedings of the ACM-SIGMOD International Conference on Management of Data,* (Inderpal Singh Mumick, ed.) pp. 469-479, 1996.
2. Hammer, J., H. Garcia-Molina, J. Widom, W. Labio, Y. Zhuge, The Stanford Data Warehousing Project, *Data Engineering Bulletin*, Vol. 18, 2, pp. 41-48, 1995.
3. Hanson, E. N., A Performance Analysis of View Materialization Strategies, *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, (Umeshwar Dayal & Irving L. Trniger, eds.), pp. 440-453, 1987.
4. Inmon, W. H., *Building the Data Warehouse* Second Edition, New York, John Wiley, 1994
5. Kimball, R., *The Data Warehouse ToolKit*, New York, John Wiley, 1996.
6. Sheth, A. P., J. A. Larson, Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases, *ACM Computing Surveys*, Vol. 22, 3, pp. 183-236, 1990.

Acknowledgment

The author wishes to thank the referees for their suggestions.