OLAP++: Powerful and Easy-to-Use Federations of OLAP and Object Databases

Junmin Gu
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA.
jgu@lbl.gov

Torben Bach Pedersen
Department of Computer Science, Aalborg University, 9220 Aalborg Ø, Denmark.
tbp@cs.auc.dk

Arie Shoshani
Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA.
shoshani@lbl.gov

Abstract

We describe the OLAP++ system for federating OLAP and object databases. The system allows users to easily pose OLAP queries that reference external object databases. This enables very flexible and fast integration of object data in OLAP systems without the need for prior physical integration.

1. Introduction

On-Line Analytical Processing (OLAP) systems provide good performance and ease-of-use when retrieving summary information from very large amounts of data. However, the complex structures and relationships inherent in related non-summary data are not handled well by OLAP systems. In contrast, object database systems are built to handle such complexity, but do not support summary querying well.

This paper presents OLAP++, a flexible, federated system that enables OLAP users to exploit simultaneously the features of OLAP and object database systems. In a previous paper [1], we have defined a comprehensive framework for handling federations of OLAP and object databases, including the SumQL++ language that allows OLAP systems to naturally support queries that refer to and retrieve data from object databases. The OLAP++ system allows data to be handled using the most appropriate data model and technology: OLAP systems for summary data and object database systems for the more complex, general data. Also, the need for physical integration of data is reduced considerably. We present a case study based on the Transaction Processing Council (TPC) TPC-R benchmark [3]. The system is implemented in C++ on top of the Object Protocol Model (OPM) system [4] and the Microsoft SQL Server OLAP Services system [2].

2. Federations of OLAP and Object Databases

OLAP systems use a *multidimensional* view of data that typically categorizes data as being measurable *facts* (measures) or *dimensions*, which are mostly textual and characterize the facts. Dimensions are structured using *categories* (levels) that correspond to the required levels of detail. Object systems use the familiar concepts of *classes*, *attributes*, and *relationships* between classes. A *federation* between an OLAP and an object database is defined by specifying a *link* between a category in the OLAP database and a class in the object database.

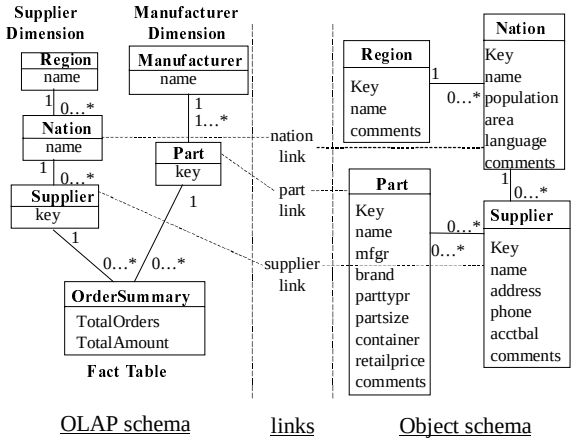


Figure 1: Schema of the Federation

Figure 1 shows an example schema of a federation in UML notation. The schema is based on the TPC-R benchmark [3], but has been divided into an OLAP part and an object part. The measured facts in the OLAP schema are the *total number of orders* and the *total cost amount* for the orders. The facts are characterized by a *Supplier dimension* and a *Manufacturer dimension*. The Supplier dimension has *Customer, Nation,* and *Region*

categories that allow the facts to be summarized to the required level of detail. The Manufacturer dimension has the categories *Part* and *Manufacturer*. The object part of the schema has *Region, Nation, Supplier,* and *Part* classes and relationships between them. Link *nationlink* connects the Nation category in the OLAP part to the Nation class in the object part as indicated by the dotted lines. Links *supplierlink* and *partlink* connect the Supplier category and class, and the Part category and class, respectively. Below is an example SumQL++ query for the schema.

SELECT TotalAmount **INTO** testdb
BY_CATEGORY Manufacturer, Nation
FROM OrderSummary
WHERE (Region = "ASIA") **AND**
Nation.nationlink.[Nation].population > 100,000,000

The above query gets the total cost amount for the two-dimensional cross product of nation and manufacturer where the nations have populations beyond 100 million and are in the Asian region. This query uses the link "nationlink" to go from the OLAP schema to the object schema. The class name in the square brackets is optional and is only specified here to indicate the class reached by going through the link.

3. System Architecture

The overall architecture of the federated system is seen in Figure 2. The object part of the system is based on the OPM tools [4] that implement the Object Data Management Group's (ODMG) object data model [5] and the Object Query Language (OQL) [5] on top of a relational DBMS, in this case the ORACLE RDBMS. The OLAP part of the system is based on Microsoft's SQL Server OLAP Services using the Multi-Dimensional eXpressions (MDX) [2] query language. The GUI is implemented as Java classes running in a standard Web browser for optimal flexibility.

When a SumQL++ query is received by the Federation Coordinator (FC), it is first parsed to identify the measures, categories, links, classes and attributes referenced in the query. Based on this, the FC then queries the metadata to get information about which databases the object data and the OLAP data reside in and which categories are linked to which classes.

Based on the object parts of the query, the FC then sends OQL queries to the object databases to retrieve the data for which the particular conditions hold true. This data is then put into a "pure" SumQL statement, i.e., without object references, as a list of category values. This SumQL statement is then sent to the OLAP database layer to retrieve the desired measures, grouped by the requested categories. The SumQL statement is translated into MDX by a separate layer, the "SumQL-to-MDX translator", and the data returned from OLAP Services is returned to the FC.

The reason for using the intermediate SumQL statements is to isolate the implementation of the OLAP data from the FC. As an another alternative, we have also implemented a translator into SQL statements against a relational "star schema" design.

The system offers good query performance even for large databases while making it possible to integrate existing OLAP data with external data in object databases in a flexible way that can adapt quickly to changing query needs.

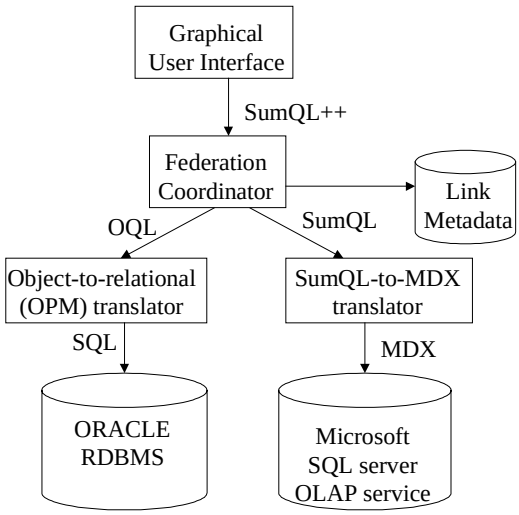


Figure 2: Architecture of the Federated System

4. The Demonstration

The demonstration will show the specification of, and query processing for, specific queries on a large TPC-R-based database. First, the use of the system will be demonstrated. Second, we will describe the details of query processing in the system. In the demonstration, we will also show how new federations can be specified "on-the-fly" and used immediately. Supporting material in the form of slides and posters will be used in the demonstration.

4.1 User Interface

The web screen interface shown in Figure 3 below shows how the user perceives the specification of a SumQL++ query. Figure 3 shows the selection of the summary measure "TotalAmount". This is followed by the section with the category attributes "Manufacturer" and "Nation". Note that each category can be selected from a "category hierarchy". In the figure, "Nation" was selected from the "Region-Nation-Supplier" category hierarchy. The order of the category grouping

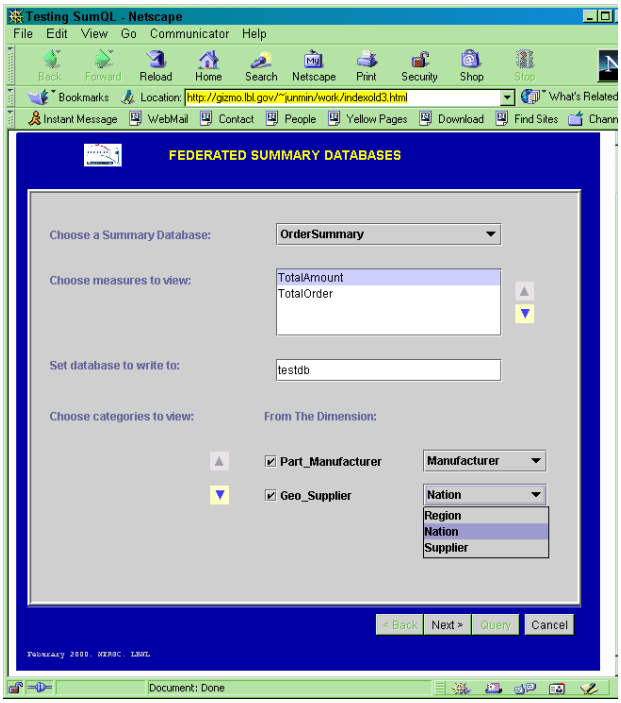


Figure 3: Selection of Measures and Categories

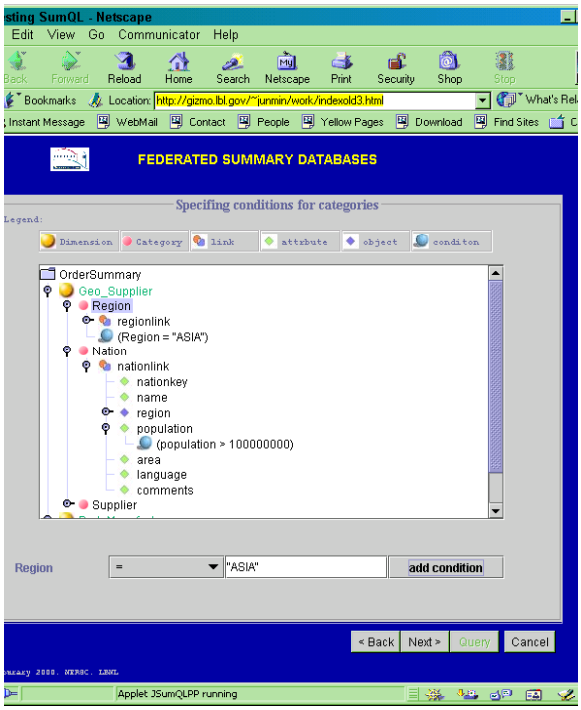


Figure 4: Specification of Query Conditions

can be specified in this screen as well by switching the dimension positions.

Figure 4 shows the specification of query conditions. Initially, each dimension is shown with its categories and links to the object database. If a category is selected, a category condition can be entered. In the figure, Region= "ASIA" was selected. If a link is clicked on, then the attributes of the object linked to are shown. The user can select an attribute to specify a condition. In the figure, the condition "population > 100 Million" was selected through the "nationlink". The result of the above selections is a concise SumQL++ query (the same query as the example in Section 1), as shown next.

SELECT TotalAmount **INTO** testdb
BY_CATEGORY Manufacture, Nation
FROM OrderSummary **WHERE** (Region = “ASIA”)
AND (Nation.nationlink.population > 100000000)

The result of this query is then displayed on the user's screen, as shown in Figure 5.

4.2 Query Processing

We now proceed to describe the steps in the query processing in more detail.

After the query is generated, the system parses the query to determine the OLAP and object parts. For the example above the result of the parsing is:

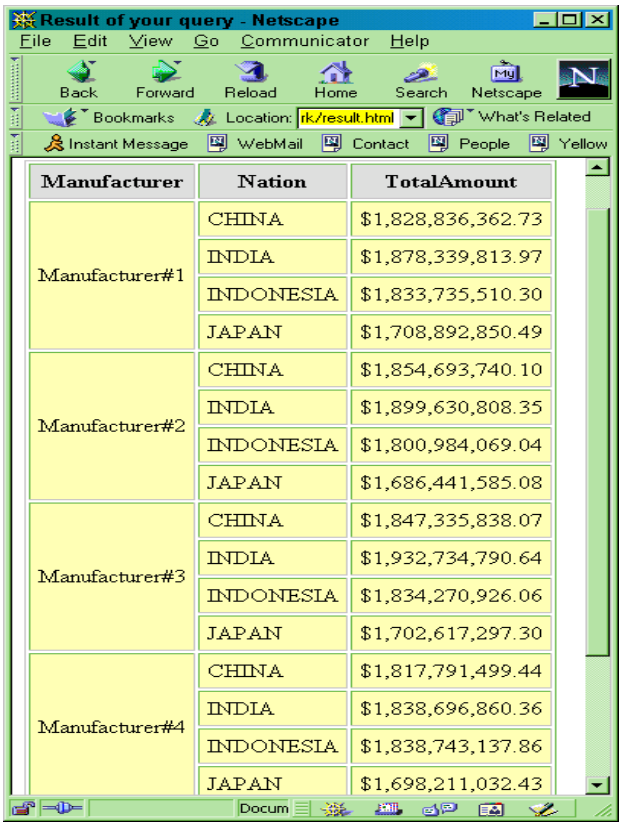
SELECT TotalAmount **INTO** testdb
BY_CATEGORY Manufacturer, Nation
FROM OrderSummary
[AND]
predicate: CATEGORY = Region
no object path
-------> = "ASIA"
predicate: CATEGORY = Nation
LINK = nationlink
PATH = .
ATTR = population
-------> > 100000000

Each link predicate is then evaluated by the object system. For example, the following OQL query is passed to the object DB system to find the nations with a population of more than 100 million:

SELECT name = @n001
FROM @n000 **IN** tpcr:NATION,
@n001 **IN** @n000.name
WHERE @n000.population > 100000000;

After the results are returned, they are used in the OLAP part of the system to generate the following SumQL query that retrieves the desired data.

SELECT TotalAmount **INTO** testdb
BY_CATEGORY Manufacturer, Nation
FROM OrderSummary **WHERE**
(Region = 'ASIA' **AND** Nation **IN** ('BRAZIL', 'INDIA', 'INDONESIA', 'JAPAN', 'CHINA', 'RUSSIA', 'UNITED STATES'))



Manufacturer	Nation	TotalAmount
Manufacturer#1	CHINA	$1,828,836,362.73
	INDIA	$1,878,339,813.97
	INDONESIA	$1,833,735,510.30
	JAPAN	$1,708,892,850.49
Manufacturer#2	CHINA	$1,854,693,740.10
	INDIA	$1,899,630,808.35
	INDONESIA	$1,800,984,069.04
	JAPAN	$1,686,441,585.08
Manufacturer#3	CHINA	$1,847,335,838.07
	INDIA	$1,932,734,790.64
	INDONESIA	$1,834,270,926.06
	JAPAN	$1,702,617,297.30
Manufacturer#4	CHINA	$1,817,791,499.44
	INDIA	$1,838,696,860.36
	INDONESIA	$1,838,743,137.86
	JAPAN	$1,698,211,032.43

Figure 5: Query Result

This, in turn, gets translated into MDX as follows.
SELECT {[Measures].[L ExtendedPrice] } **ON COLUMNS, INTERSECT**
(**CROSSJOIN**([Part_Manufacture].[P Mfgr].**MEMBERS**, **DESCENDANTS**([R Region Name].[ASIA],[N Nation Name],**SELF**)),
CROSSJOIN([Part_Manufacture].[P Mfgr].**MEMBERS**, {[N Nation Name].[BRAZIL],[N Nation Name].[INDIA],[N Nation Name].[INDONESIA],[N Nation Name].[JAPAN],[N Nation Name].[CHINA],[N Nation Name].[RUSSIA],[N Nation Name].[UNITED STATES]})) **ON ROWS FROM** OrderSummary

The result is then stored in the Oracle database "testdb," to make it available for further processing, and converted to HTML for presentation to the user.

This section was intended to illustrate the amount of work that a user will have to go through without the aid of the user interface and the federated translation tools. In particular, we wish to emphasize the usefulness of the OLAP-object database links to generate the combined result. Also, the users are spared the verbosity of MDX (which is hidden from them). It is optional to display the concise SumQL++ expression to the user, as a way to verify the correctness of the query. Due to space constraints, we do not describe the specification of new links in this paper. However, this will be shown at the demonstration.

References

1. T. B. Pedersen, A. Shoshani, J. Gu, and C. S. Jensen. Extending OLAP Querying to External Object Databases. Submitted for publication.
2. Microsoft Corporation, OLE DB for OLAP Version 1.0 Specification. *Microsoft Technical Document*, 1998.
3. Transaction Processing Council. The TPC-R Benchmark. *URL: <www.tpc.org>*. Current as of June 1, 2000.
4. I-Min A. Chen, Victor M. Markowitz: An Overview of the Object-Protocol Model (OPM) and OPM Data Management Tools. *Information Systems* 20(5): 393-418 (1995).
5. R. G. G. Cattell et al. (editors). *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, 1997.