

XPERANTO: A Middleware for Publishing Object-Relational Data as XML Documents

Michael Carey* Jerry Kiernan Jayavel Shanmugasundaram⁺
Eugene Shekita Subbu Subramanian

IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120

carey@acm.org, kiernan@almaden.ibm.com, jai@cs.wisc.edu,
shekita@almaden.ibm.com, subbu@us.ibm.com

Abstract

The eXtended Markup Language (XML) is quickly emerging as the universal format for publishing and exchanging data on the World Wide Web. As a result, data sources, including object-relational databases, are now faced with a new class of users and applications; customers and programs that would like to deal directly with XML data rather than being forced to deal with the data source's particular (e.g., object-relational) schema and query language. The goal of the XPERANTO middleware project is to support this new class of users and applications. XPERANTO does this by providing query-able "XML Views" over the underlying object-relational database. Users can then query and (re)structure XML data using an XML query language, without having to deal with the underlying SQL tables and query language. The XPERANTO system translates XML-based queries into SQL requests, receives the tabular

query results, converts them into XML, and then returns XML documents to the system's users and applications.

1. Motivation and Introduction

XML [2][4] is emerging as the standard for publishing and exchanging data on the World Wide Web. As a result, many "e-businesses" want to publish their existing data as XML documents so that their business partners can process them. For example, an online wholesale bookstore may want to make its inventory available in XML form so that retail merchants can process this information over the Internet. It is also important to provide query capability over such XML data because, for instance, a retail merchant may want to check whether the wholesale bookstore has a particular book in stock (this is a selection query over the inventory XML document).

The goal of the XPERANTO project [3] at the IBM Almaden Research Center is to publish *object-relational data* as XML documents. XPERANTO operates as a middleware on top of existing (object-)relational database systems and provides query-able "XML Views" over them. In doing so, it shields users of XPERANTO from the underlying database and language (SQL) and allows them to create and query XML data using an XML query language (currently XML-QL [5]). Thus, businesses that have large amounts of important business data stored in existing (object-)relational databases can use XPERANTO to make this data available on the Internet in query-able XML form.

XPERANTO differs from other similar systems that we are aware of [6] in the following ways. First, XPERANTO is based on a "pure XML" philosophy – users and developers do no need to know or learn SQL.

*Currently at Propel, 2350 Mission College Blvd., Santa Clara, CA 95054.

⁺Also at the University of Wisconsin, Madison, WI 53706.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egypt, 2000

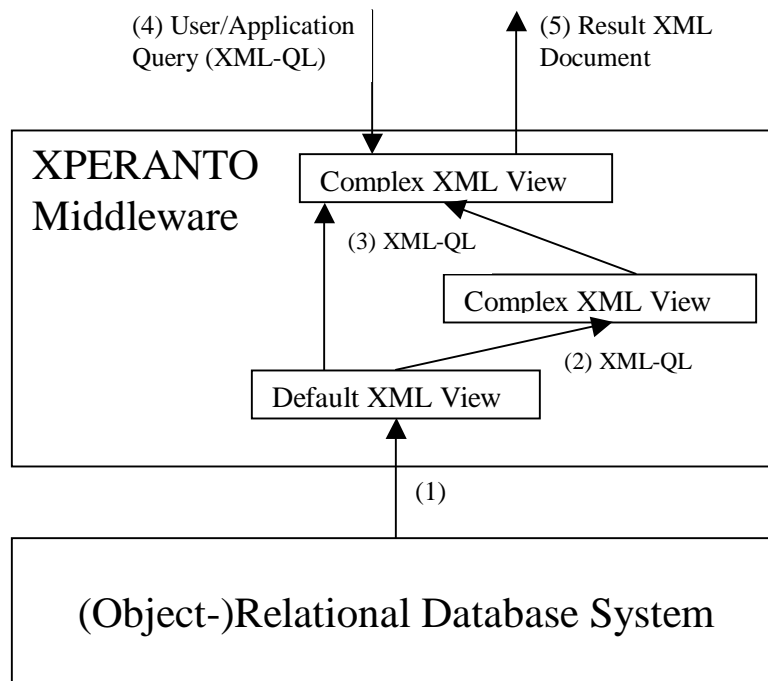


Figure 1: XPERANTO Architecture

Second, XPERANTO attempts to push as much processing down to the relational engine as possible for more efficient query execution [7]. Third, XPERANTO handles object-relational structures in addition to flat relational structures. Finally, XPERANTO enables the seamless querying of relational data (such as column values) and meta-data (such as column names).

For the remainder of this demo proposal, we outline the architecture of the XPERANTO system and show it can efficiently support query-able XML views by doing smart XML query rewrites and by harnessing the sophisticated processing capability of the underlying (object-)relational engine.

2. The XPERANTO High Level Architecture

As mentioned earlier, XPERANTO is a middleware system that works on top of existing (object-)relational database systems. Figure 1 shows the high level architecture of the XPERANTO system. XPERANTO starts by providing a default virtual view of a given (object-)relational database (step 1). XML application developers can then create more complex or specialized views (which are also virtual) on top of this default view (step 2) by using an XML query language (currently XML-QL). These tailored views can then be made available to other applications and/or user groups, who can in turn create more complex view over the existing views (step 3). The views can then be queried using the same XML query language used to define them (step 4) and XPERANTO returns the result XML document (step 5). XPERANTO clients use an XML-based RPC facility –

Simple Object Access Protocol [1] – to communicate with the XPERANTO middleware for creating and querying XML views and retrieving the XML results.

Note that XPERANTO takes a "pure XML" approach to the problem of publishing (object-)relational data as XML - i.e., developers interact with XPERANTO using only XML (and an XML query language); they do not have to use or learn SQL.

An interesting and novel aspect of XPERANTO is that it provides a uniform framework that allows users to use XML as a way to seamlessly query over both relational data and relational metadata. This power stems from the fact that the XPERANTO default XML view of a relational database does not distinguish between data (such as column values) and metadata (such as column names). In this sense, XPERANTO provides users and applications a more powerful query capability than SQL.

3. XPERANTO Internals

We now describe the internals of the XPERANTO system and show how it supports query-able XML views over relational sources. As Figure 2 shows, when an user or applications poses an XML-QL query over an XML view, it is first parsed and translated into an internal representation called XQGM (Xml Query Graph Model). The representation is designed to be language neutral (i.e., not tied to the XML-QL query language) so that XPERANTO can be easily modified to support the standard XML query language when one becomes available.

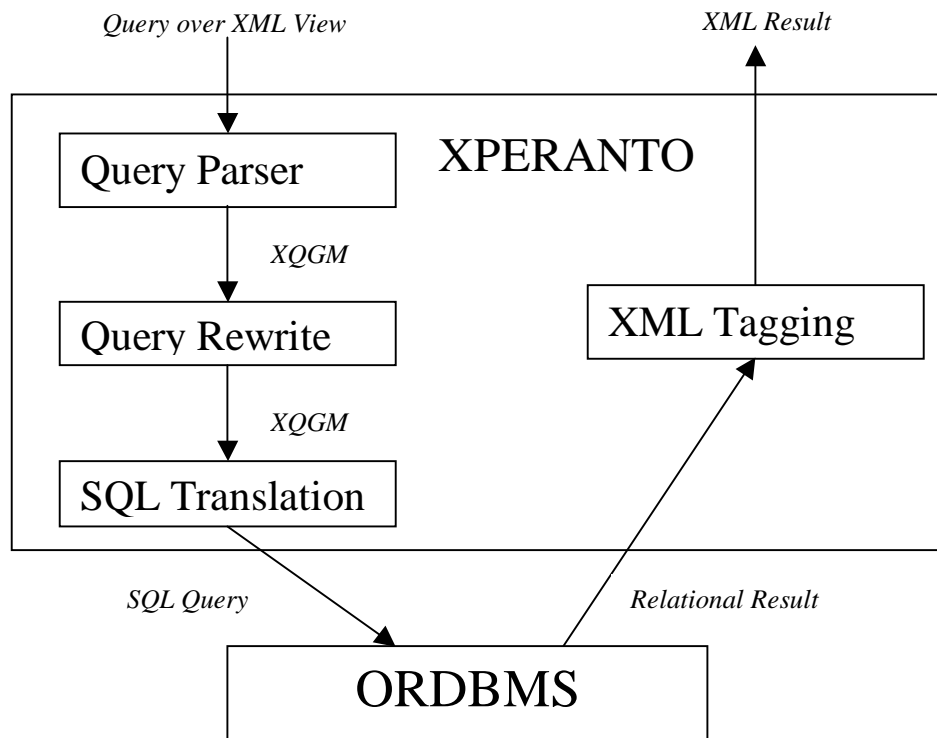


Figure 2: XPERANTO Internals

Once the query parser creates XQGM for a user query, the XPERANTO rewrite engine performs XML view composition in the middleware layer (i.e. outside of any database system) and produces simplified XQGM that is semantically equivalent to the original XQGM. The SQL generator component then converts this simplified XQGM representation to SQL queries that generate the content of the XML document. A typical XML query over an XML view will result in just one SQL query being issued, thus harnessing the query processing capabilities of the underlying relational engine [7]. The relational query results are then tagged in the XPERANTO middleware layer in order to produce the result XML document.

One attractive aspect of the XPERANTO approach is that it can work with any existing relational database system since the XPERANTO system generates regular SQL and tags the results outside the database engine.

4. The Demo

In this demo, we will illustrate the "pure XML" approach by presenting how the XPERANTO system automatically creates a default XML view over an (object-)relational database system. We will then show how developers can create more complex XML views on top of the system's default view. We will demonstrate how views are composed on top of other views, how queries on these views are translated into SQL queries, and how the results are converted into XML in the XPERANTO middleware layer.

5. References

- [1] T. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. Nielson, S. Thatte, D. Winer, "SOAP: Simple Object Access Protocol," <http://www.w3.org/TR/SOAP>.
- [2] T. Bray, J. Paoli, C.M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", <http://www.w3.org/XML/1998/06/xmlspec-report-19980910.htm>.
- [3] M. Carey, D. Florescu, Z. Ives, Y. Lu, J. Shanmugasundaram, E. Shekita, S. Subramanian, "XPERANTO: Publishing Object-Relational Data as XML," *Workshop on the Web and Databases (Informal Proceedings), May 2000*.
- [4] R. Cover, "The SGML/XML Web Page," <http://www.oasis-open.org/cover/xml.html>.
- [5] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, D. Suciu, "XML-QL: A Query Language for XML," *Proceedings of the 8th International World Wide Web Conference, Toronto, May 1999*.
- [6] M. Fernandez, W. Tan, D. Suciu, "SilkRoute: Trading Between Relations and XML," *Proceedings of the 9th International World Wide Web Conference, Amsterdam, May 2000*.
- [7] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, B. Reinwald, "Efficiently Publishing Relational Data as XML Documents", *Proceedings of the VLDB Conference, Egypt, September 2000*.