Using Views to Query XML Documents

Mario Cannataro  
*Magna Graecia University of Catanzaro, Italy*

Sophie Cluet  
*INRIA Rocquencourt-Xyleme S.A., France*

Giuseppe Tradigo  
*Magna Graecia University of Catanzaro, Italy*

Pierangelo Veltri  
*Magna Graecia University of Catanzaro, Italy*

Dan Vodislav  
*CNAM Cedric-Lab, Paris, France*

**INTRODUCTION**

We consider using views in databases and in particular their applications to access heterogeneous and large volumes of digital documents, focusing on (1) views in databases and (2) views for data integration.

We start our paper first answering the questions what is a view and what are their applications. We give a short representation of this concept and reference Ullman (1988) and Abiteboul, Hull, and Vianu (1995) for more detailed and formal definitions in the database literature.

Views are used for several reasons to manipulate data coming from different sources through a unique and more convenient structure; e.g., to hide data (e.g., for security reasons) or to restructure data, giving a simplified vision of the database.

Information may be grouped in one or more collections of data depending on the database model (e.g., as a set of tables or classes in a structured model or in a graph in a semi-structured model). View facilities allow one to create an imaginary source collecting interesting data. For instance, in the case of a relational database, a view might be an imaginary table containing data coming from several tables of the database. Once views are created they can be queried by users as if they were part of the database.

Views can be defined, queried, and modified using view definition, view query, and view manipulation languages. These three languages depend on the data model; e.g., for a relational data model, the view definition and query languages coincide with SQL, the standard query language to access to databases.

A view is created specifying a schema, a domain, and a view definition.

- **The view domain** is a set of sources containing information that the view represents.
- **The view schema** is the structure of the view and describes how data are presented and are queried by the users.
- **The view definition** expresses the links existing between data in the sources (actual data) and data in the view. It is expressed using the view definition language (see Figure 1).

In a simple case, the view domain is a set of data sources in the same database (e.g., a set of tables in a relational database), and the view definition language is that of the database. In more complex cases, the view

![Figure 1. A view on a relational database](image-url)
domain is a set of heterogeneous sources. The view schema provides a uniform access to the domain, but each source must be linked to the view schema by view definitions that depend on the source. Although there are some variations, the principle of using a view mechanism is to simplify the access to data sources. Users querying a view schema will obtain results as if they were querying one simple database. View engines, implemented in database management systems, are in charge of translating the query against the view into queries against the data.

Once defined, views can be virtual or materialized, depending on the way data are stored in the data structure associated with the view (materialized) or if the view is just a virtual representation (portion) of data sources (virtual views or, simply, views). Materializing a view consists of computing the associated query once and storing its result. Querying a virtual view implies a translation of the query using view definition to retrieve data on sources. Virtual views are used when data are updated frequently on data sources and when data cannot be replicated.

Views are materialized typically to improve query performance (Goldstein & Larson, 2001) and data availability. Querying a materialized view is much faster than querying a virtual one: A query against a materialized view is like a query against an actual database. The speed difference may be critical in applications where the query rate is high and the view is complex (e.g., with aggregate functions or joining remote sources; Chaudhuri, Krishnamurthy, Potamianos, & Shim, 1995; Srivastava, Dar, Jagadish, & Levy, 1996; Zaharioudakis, Cochrane, Lapis, Pirahesh, & Urata, 2000). Materialized views are used in data warehouses (Hammer, Garcia-Molina, Widom, Labio, & Zhuge, 1995) and distributed and multidimensional databases. Nevertheless, using materialized views implies new problems such as view selection, i.e., which views to materialize, and view maintenance, i.e., refreshing data into the view when data change (Griffin & Libkin, 1995).

 Views can also be used for data integration. A data integration system provides a uniform query interface to a multitude of autonomous, distributed, and heterogeneous data sources, which may reside within a local database (e.g., database of an enterprise) or on distributed databases. The idea is to free the user from having to deal with several sources to have a complete answer to his queries. Queries are formulated using a uniform query language against a uniform virtual view of the data (also known as a mediator) instead of querying different data schemas. The querying system translates the query against the view into queries against the sources, retrieves results on data sources, and finally presents them to the user (Lenzerini, 2002).

Data integration systems represent a good solution to the information retrieval problem such as a search of the Web. Users interested in finding information over the full Web use key-word-based search engines such as Altavista or Google. Such methods are often not precise enough and return a lot of useless URLs. Answering precise queries such as “find the names and addresses of Spanish museums that own a painting by Picasso” requires a lot of tedious browsing through useless pages.

There have been many research projects focusing on data integration for structured and semi-structured data sources. The common idea is to provide a global view as an interface between end users and data sources. A query is formulated on the global view and then translated in a union of queries against the different sources. The research projects can be distinguished according to the way global view and sources are connected. In a first approach called global-as-view (GAV), the global view is a collection of views defined over the schema of the local sources (Cluet, Simeon, & Delobel, 1998). A query translation algorithm can be very efficient, but the global view has to be changed whenever a local source is updated. In a second approach called local-as-view (LAV), local sources are defined as a view over the global view. Recently, a mixed approach known as GLAV has been proposed (Lenzerini, 2002).

The emergence of the XML language (W3C, 2000) as the new (and global) standard for data exchange on the Web may change things positively by adding structure where there was none. Many research projects have proposed XML as a language to express a common interface between existing databases or to integrate in a uniform view heterogeneous data sources (Manolescu, Florescu, & Kossman, 2001). Other database-like applications, such as data monitoring, document changing notifications, active databases, and electronic commerce, have taken or will take advantage of the XML technology.

This paper presents views and database models and reports the proposals of a view system presented in Aguilera, Cluet, Milo, Veltri, and Vodislav (2002) and in Veltri (2002) as a system to query large-scale heterogeneous XML data.

BACKGROUND

View definitions depend on the database model. In relational databases (Ullman, 1998), data is organized in tables (relations), each one with a schema which defines its structure (i.e., attributes and their base types such as integer, string, etc). The instance of a relation consists of a set of tuples, each containing attributes instances of the appropriate base types. For instance, the following set of relations contains data on employees and projects and belongs to the relational database of an enterprise:
Related Content

**Semantic Multigranularity Locking for Object-Oriented Database Management Systems**
[www.igi-global.com/article/semantic-multigranularity-locking-object-oriented/51178?camid=4v1a](www.igi-global.com/article/semantic-multigranularity-locking-object-oriented/51178?camid=4v1a)

**Spatio-Temporal Indexing Techniques**
[www.igi-global.com/chapter/spatio-temporal-indexing-techniques/20710?camid=4v1a](www.igi-global.com/chapter/spatio-temporal-indexing-techniques/20710?camid=4v1a)

**Multilevel Databases**
[www.igi-global.com/chapter/multilevel-databases/11177?camid=4v1a](www.igi-global.com/chapter/multilevel-databases/11177?camid=4v1a)

**Empirical Comparison of 3-D Virtual World and Face-to-Face Classroom for Higher Education**
[www.igi-global.com/article/empirical-comparison-virtual-world-face/74702?camid=4v1a](www.igi-global.com/article/empirical-comparison-virtual-world-face/74702?camid=4v1a)