Enabling End-users to Construct Data-intensive Web-sites from XML Repositories: An Example-based Approach

Atsuyuki Morishima Seiichi Koizumi Hiroyuki Kitagawa Satoshi Takano
Shibaura Institute of Technology University of Tsukuba
Saitama, Saitama, Japan Tsukuba, Ibaraki, Japan
{mori, izumi, kitagawa, ppoi}@kde.is.tsukuba.ac.jp

1 Introduction

We demonstrate how our system can help users construct data-intensive Web sites from XML repositories and databases in a very intuitive fashion. The key idea is to show the system some *example operations with data instances* and let it infer the operations for the whole collection of data objects in the data repositories. This problem is challenging, because we allow data in the data repositories to be semistructured, and inferring (generalizing) the operations is a non-trivial task. We already found through experiments that even users without any knowledge on database-related concepts (such as database schemas, SQL-style queries and path expressions) can construct fairly complex Web-sites from a collection of XML documents and a relational database, by following their intuitions. We think our system is an example of *how end users can be empowered by the semistructured data technologies* the research community has been developing, because the Web-site construction process is realized by them.

The demonstration is done with an implemented system of our AQUA (Amalgamation of QUerying and Authoring) project[5] . The project started with a recognition that the wide acceptance of the world wide web is not only due to its easy way to access data but also to the simple mechanism that enables end users to publish their own, personal Web contents[3]. Some of such personal Web sites are very data-intensive: For example, movie enthusiasts manage a lot of movie information, and researchers' Web sites have their publications. (They are not necessarily computer science researchers.) While they write tons of HTML documents, few of them use database management systems. We attribute it to the following two causes: (1) For end users, incorporation of databases' contents into Web sites is difficult: They can use commercially available authoring tools with drag-and-drop facilities to design their sites. However, they are typically required to write queries in SQL, which is a completely different framework from the page authoring. (2) The nature of live information from the real world

**Proceedings of the 27th VLDB Conference,
Roma, Italy, 2001**

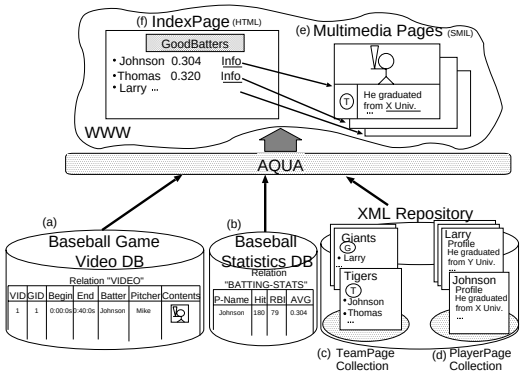

Figure 1: A Web-site on top of information sources

and users' lack of knowledge on database principles tend to cause frequent changes on the data structures and result in semistructured data. In such situations, it is troublesome to use relational databases for data management. Based on the observation, we designed our system which (1) requires no specification of SQL-style queries in data manipulation languages, and (2) allows semistructured data repositories (such as XML document collections) to be used as underlying information sources.

2 Example Scenario

We consider the following three information sources. (1) *A baseball game video database* (Figure 1(a)): This is a relational database which contains video objects (such as mpeg objects) of batting scenes and their metadata. (2) *A baseball statistics database* (Figure 1(b)): This is a relational database which maintains the latest statistics about baseball players. (3) *A baseball players' profile XML repository*: This repository contains the profile information of baseball players. It has two collections of documents: TeamPages (Figure 1(c)) and PlayerPages (Figure 1(d)). Each TeamPage contains the team logo (as a reference to an image file) and its player names. Each PlayerPage contains the profile data. We also assume that the pages are semistructured (shown in the demonstration).

The Web-site we construct here is shown in Figure 1. A SMIL Web page is constructed for each player whose hitting average is more than 0.3 for the current season. It is a *multimedia page* (Figure 1(e)), which consists of three components: (1) *A sequential rendering of scenes (video objects)* in which he is at bat. (2) *The logo of the team he belongs to.* (3) *Text description of his profile.* An index

HTML page is also created (Figure 1(f)). It contains the selected players' names, hitting averages, and links to the players' multimedia pages.

3 Using AQUA System

AQUA looks like just a common authoring tool for HTML and SMIL pages (Figure 2). Users are only required to drag and drop data objects presented in windows named *DataBoxes* (Figure 2(a)-(d)) into a blank window named the *Canvas* (Figure 2(e)(f)). Manipulation of a collection of data objects in the underlying data repositories is done by designating an existing data object as an *example*. Then, the data object (the example) serves as the representative of a set of data objects. A drag-and-drop operation of the example is interpreted as manipulation of the set of data objects. Therefore, the object-at-a-time authoring framework and the set-at-a-time data manipulation (querying and restructuring) framework are integrated in a seamless way.

What makes the problem challenging is that AQUA allows data in the repositories to be semistructured, and so the understanding of the user's intention is a non-trivial task. What is the set of objects an example object represents? How those objects are related to each other? Internally, AQUA is based on well-known techniques developed for semistructured data query languages, such as a tree-style data modeling scheme and path regular expressions to qualify data objects. Our technical contribution is development of an algorithm, which infers semistructured data queries from users' interactive operation with the system.

Figure 2: User Interface

4 System Architecture

Figure 3 shows the architecture of our demonstration system. The heart of the system is the AQUA interface part. Users communicate with the system through GUI components, namely, *DataBox* and *Canvas*. DataBoxes are windows to show data objects in underlying information sources. Users drag-and-drop the data objects from the DataBoxes to the Canvas in order to show the system example Web pages. The DataBox Manager produces *matching patterns* and *selection/join conditions* against the underlying data, based on users' interaction with DataBoxes. The Canvas Manager produces *grouping information* and *site template*. The matching patterns, selection/join conditions

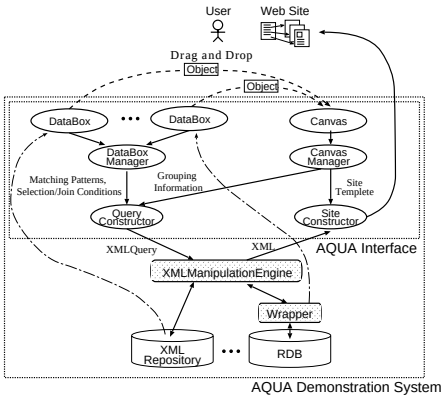

Figure 3: Demonstration System Architecture

and grouping information are fed into the *query constructor* to produce a semistructured (XML) data query. Although our demonstration system uses a proprietary XML manipulation engine, we believe the AQUA interface can work with other data manipulation systems with appropriate translators. The *site constructor* embeds the query result from the XML manipulation engine into the site template and produces the result Web-site.

5 Related Work

A number of declarative Web-site management systems proposed can deal with semistructured data. Some of them are surveyed in [4]. AQUA can contribute as a complement to them in the sense that it provides users with an intuitive user interface for semistructured Web-site management systems.

AQUA can be used as an XML graphical query environment. While other graphical query languages such as XML-GL[2] require users to *specify* queries, AQUA *infers* queries from instance-based example operations.

Many results in the area of computational learning theory[1] should be applicable to development of improved algorithms for AQUA. Introduction of *negative examples* and various techniques for *active learning* is one of the improvements we are planning.

References

[1] D. Angluin. Computational Learning Theory: Survey and Selected Bibliography. *Proc. 24th Annual ACM Symposium on Theory of Computing*, pp. 351-369, 1992.

[2] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A Graphical Language for Querying and Restructuring XML Documents. *WWW8 / Computer Networks*, Vol. 31, No. 11-16, pp. 1171 - 1187, 1999.

[3] T. Erickson. The World-Wide-Web as Social Hypertext. *Commun. ACM*, Vol. 39, No.1, pp. 15 - 17, 1996.

[4] P. Fraternali. Tools and Approaches for Developing Data-intensive Web Applications: A Survey. *ACM Comput. Surv.*, Vol 31, No.3, pp. 227 - 263, 1999.

[5] A. Morishima, S. Koizumi, and H. Kitagawa. Drag and Drop: Amalgamation of Authoring, Querying, and Restructuring for Multimedia View Construction. *Proc. 5th IFIP 2.6 Working Conference on Visual Database Systems(VDB5)*, pp. 257 - 276, 2000.