Visual SQL-X: A Graphical Tool for Producing XML Documents from Relational Databases

Renzo Orsini Massimo Pagotto
Dipartimento di Informatica, Università di Venezia
Via Torino, 153 – 30173, Mestre
Italy
{orsini, pagotto}@dsi.unive.it

ABSTRACT

The Visual SQL-X system is presented to generate arbitrarily complex XML documents from a graphical query on a relational database. The query describes the structure of the resulting document through a tree-like representation of its structure. The system generates the XML document through the synthesis of a SQL query and a "summarization" algorithm on the resulting table.

Keywords

Visual SQL-X, XML, Visual Query Languages.

1. INTRODUCTION

An increasing need is emerging in the development of XML-enabled systems for tools which allow a fast and simple conversion of data from various sources into XML formats. For the vast majority of cases, such data come from existing relational databases. Academic researchers and software companies are actively addressing such a need, mostly with the design of query languages or extraction languages from relational databases.

At the University of Venice, the Visual SQL-X tool has been developed to produce, with a graphical interface, a query against a relational database which generates an XML document of arbitrary complexity. The novelty of the approach is that the user is not required to learn the syntax and semantics of yet another query language, but to use a simple interface which allows the construction of a tree, which defines the structure of the expected result.

2. THE TREE MODEL OF A SQL-X QUERY

The language SQL-X [1] is an extension of SQL which provides a set of operators to compose queries against a relational database, which produce complex XML documents, with a style reminiscent of report generation languages.

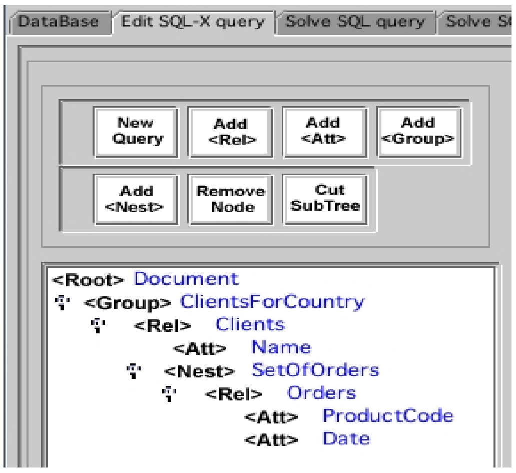
A query is represented by a tree with the following kind of nodes (for an example, see fig. 1):

1. <Root> is the tree root, and represents the whole document, containing a set of elements corresponding either to tuples or to group of tuples.
2. <Rel> represent a database relation (obtained in general through an SQL query), whose tuples are converted into elements of its immediate container.
3. <Att> (child of <Rel>), represents a column, a value of which is used as element of its container,
4. <Nest> represents the tuples of a relation which are associated, with a join operation, to a tuple of its container, and which will become roots of subtrees.
5. <Group> represents the grouping of the tuples of a child node by some expression: each group is an element containing the tuples of the group as elements.

Moreover, the user can specify if tuple fields are converted to attributes, instead of elements, the ordering of sequences, as well as other details of the conversion process.

3. OVERVIEW OF THE SYSTEM

After selecting a database, which is then analyzed to collect its metadata, the user is presented with the query editor.



```
<Root> Document
  <Group> ClientsForCountry
    <Rel> Clients
      <Att> Name
      <Nest> SetOfOrders
        <Rel> Orders
          <Att> ProductCode
          <Att> Date
```

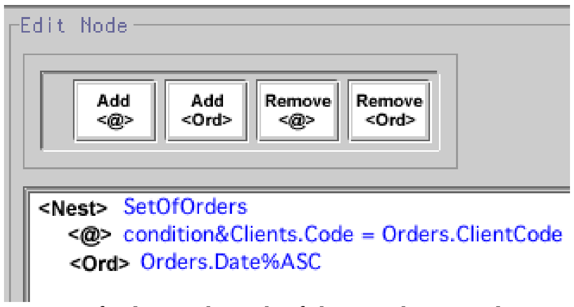
Fig. 1 The query editor

The panel shown in fig.1 allows the construction of the query tree, by selecting a node and then applying an operator. In this case, the tree represents a query which returns a set of clients. They are grouped by country, and each client contains an element with its name, and another one with the sequence of the product code and date of its orders. When a node is selected, the right panel shows the associated information, which depends on the node:

- for <Root>, the ordering of its elements,
- for <Rel>, the fields which are the elements' attributes,
- for <Nest>, the join condition, the ordering of its elements, and possibly other conditions on the tuples corresponding to the subelements,
- for <Group>, the grouping condition (e.g. the field 'Country' of 'Clients'), an ordering for its subelements, and the group's element attributes.

For instance, the definition of the node corresponding to the set of orders for each client is shown in fig.2.

Fig.2. Editing a node



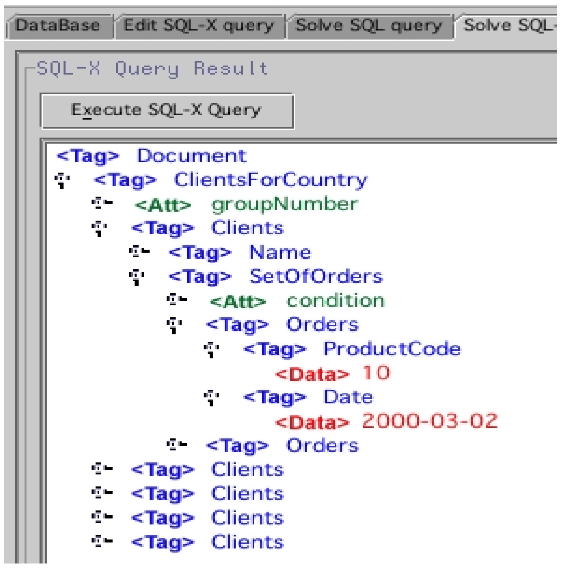
```
<Nest> SetOfOrders
   <@> condition&Clients.Code = Orders.ClientCode
   <Ord> Orders.Date%ASC
```

The system facilitates the task of the user by providing a set of panels for composing conditions and other expressions (e.g. with aggregation functions) during the construction of the tree.

In the prototype, the user can follow, with a set of panels, all the phases of the evaluation of the query: the conversion in SQL and the resulting relation, the tree which is the result of the data extraction, and the final document. For instance, the query of the example is translated into the following SQL query:

SELECT Clients.Name, Orders.ProductCode, Orders.Date
FROM Clients, Orders
WHERE (Clients.Code = Orders.ClientCode)

while in fig.3 the structure of the resulting XML document is shown (only a few elements are expanded).



```
<Tag> Document
  <Tag> ClientsForCountry
    <Att> groupNumber
    <Tag> Clients
      <Tag> Name
      <Tag> SetOfOrders
        <Att> condition
        <Tag> Orders
          <Tag> ProductCode
            <Data> 10
          <Tag> Date
            <Data> 2000-03-02
        <Tag> Orders
    <Tag> Clients
    <Tag> Clients
    <Tag> Clients
    <Tag> Clients
```

Figure 3 Resulting document

In any phase of the evaluation, the user can go back and change its definition, for instance to experiment different grouping and nesting strategies. For the lack of space, the final documents, together with its DTD, is not shown here.

3.1 Implementation

The approach taken in translating the query into SQL is that of collecting all the necessary data into a single relation, which is then read only once for producing the resulting XML document. The example previously shown is in effect a very simple query: all the work is made in the final phase, which, through a visit of the query definition tree, generates an intermediate tree containing all the data, which can be directly mapped to the DOM representation of the document. (the details of the approach can be found in a forthcoming paper [2]).

4. USE OF VISUAL SQL-X

The main use of the system is envisioned in generating programs which extract automatically XML documents from a relational database, with a simple and friendly interface.

A typical use could be for generating XML pages to be published on the Web, or transferred to some other application: many tools currently available are limited by the fact that the queries which get the data can be expressed in SQL, so that the result is always “flat”: this leads to complex programming if there is a need for structuring the data inside the page. With our tools, SQL programmers could easily provide complex content without resorting to a specialized (and not so easy) new query language.

5. RELATED WORKS

Of the many languages proposed to query XML, very few of them take a visual approach. An example is [3], which allows the direct manipulation of a tree representation of the documents’ DTD. Our approach, in contrast, attempts to build a bridge between the relational data model and the XML data model, with an approach similar to that, for instance, to that of SilkRoute [4] and PENELOPE[5], which, on the other hand, provide the user only with textual languages.

6. CONCLUSIONS AND FUTURE WORK

We are now experimenting the prototype on real cases, and trying different strategies to evaluate the query. A symmetrical tool is being developed which allows graphically the population of database relations with data extracted from XML documents, with an analogous approach.

This system is intended to be one of a set of tools in a workbench for data exchange, through XML, among different data sources and applications, developed inside the framework of the Data-X project (home page: http://www.difa.unibas.it/dataX).

7. ACKNOWLEDGMENTS

This work has been carried in the Data-X Project, sponsored by the Italian MURST.

8. REFERENCES

[1] R. Orsini, A Preliminary Proposal for SQL-X, A Language to Extract XML Documents from Relational Databases, *SEBD* 5, L’Aquila, Italy, June 2000.

[2] R. Orsini and M.Pagotto. Visual SQL-X: The system and its implementation. University of Venice, Technical Report, 2001.

[3] K. Munroe, Y. Papakonstantinou, BBQ: A Visual Interface for Browsing and Querying XML, *in Visual Database Systems* (*VBD* 2000).

[4] D. Suciu, M.Fernandez, and Wang-Chiew Tan. SilkRoute: Trading between relations and XML. *In Proc. 9th World Wide Web Conference*, 2000

[5] P. Atzeni, G. Meccca and P. Merialdo, To weawe the web, in *Proc. 23rd Int. Conf. on VLDB* (*VLDB*’97), 1997.