XML-Based Information Mediation with MIX

Chaitan Baru Amarnath Gupta Bertram Ludäscher Richard Marciano
Yannis Papakonstantinou Pavel Velikhov Vincent Chu
University of California, San Diego, La Jolla, CA 92093
{baru,gupta,ludaesch,marciano}@sdsc.edu {yannis,pvelikho}@cs.ucsd.edu

1 Background

The MIX mediator system, MIX*m*, is developed as part of the MIX Project at the San Diego Supercomputer Center, and the University of California, San Diego.[1] MIX*m* uses XML as the common model for data exchange. Mediator views are expressed in XMAS (***XML M**atching **A**nd **S**tructuring Language*), a declarative XML query language. To facilitate user-friendly query formulation and for optimization purposes, MIX*m* employs XML DTDs as a structural description (in effect, a "schema") of the exchanged data. The novel features of the system include:

- Data exchange and integration solely relies on XML, i.e., instance and schema information is represented by XML documents and XML DTDs, respectively. XML queries are denoted in XMAS, which builds upon ideas of languages like XML-QL, MSL, Yat, and UnQL. Additionally, XMAS features powerful grouping and order constructs for generating new integrated XML "objects" from existing ones.
- The graphical user interface BBQ (***B**lended **B**rowsing and **Q**uerying*) is driven by the mediator view DTD and integrates browsing and querying of XML data. Complex queries can be constructed in an intuitive way, resembling QBE. Due to the nested nature of XML data and DTDs, BBQ provides graphical means to specify the nesting and grouping of query results.
- Query evaluation can be demand-driven, i.e., by the user's navigation into the mediated view.

[1] www.npaci.edu/DICE/MIX/ and www.db.ucsd.edu/Projects/MIX/

2 MIX Architecture

The graphical user interface BBQ allows the construction of queries in an intuitive way, based on the DTDs of the mediator view (Fig. 1). The XML answer document can be browsed through a DOM-like API. The main module of MIX*m* is the *query processor*, which resolves the client queries with the mediator view definitions, resulting in a set of unfolded queries against the underlying sources. Where necessary, wrappers are used on top of sources. XMAS queries can be simplified based on the given or inferred DTDs. Evaluation and further optimization of XMAS queries is accomplished by translating them into the *XMAS algebra*. Like in TSIMMIS, query evaluation of MIX*m* is *lazy*. Another novel feature which is currently being incorporated into the system is DOM-VXD (DOM for Virtual XML documents), where query evaluation is driven by the client's navigation into the virtual XML view.

3 Mediation Example

We illustrate the main features by means of a typical mediation application which we refer to as the *home buyer's scenario*.

Home Buyer's Scenario. A user who wants to buy a home wants to make use of information available from the Web to guide this decision. A possible query that the user may issue is:

"Find all houses with 3 bedrooms, 2 baths, interior area at least 1600 sq.ft., priced between \$250k and \$350k, in regions where the school rating is at least 70 (out of 100) and the crime rate is no more than 15 incidents per year. Group the answers by region and order them by price. For each home also show the nearby schools."

To facilitate such a query, XML wrappers for the different Web sources have to be provided. In our case, this involves sources HOME, SCHOOL, SCHOOL-DISTRICT, and CRIME.[2] Using BBQ, a query like the one in Fig. 3 is eventually created.

[2] See www.realtor.com, www.ads.com, homeadvisor.msn.com,

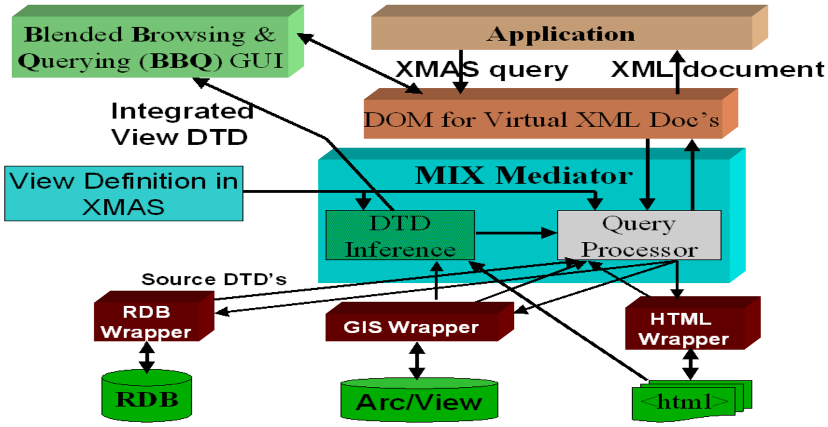

Figure 1: MIX architecture

BBQ in a Nutshell. BBQ allows to define complex XML queries in an elegant and intuitive way, based on the view DTDs provided by the mediator. Fig. 2 depicts the main windows of BBQ: The *condition* and *answer windows* are used for query composition, and define the *head* and the *body* of the generated XMAS query, respectively. The results returned via MIX*m* are displayed in a separate window and can be browsed in the usual way. Since the answer is itself a (possibly virtual) XML document, it can also be queried using the same interface. In this way, BBQ achieves its goal of smoothly integrating browsing and querying of XML data.

Initially, the condition window shows the root element `home_buyer` of the mediator view (Fig. 2). Now the user can successively click on subelements of `home_buyer` thereby exposing as much of the DTD structure as necessary. By checking an element, the user specifies that the corresponding element must be present in the data. Additional constraints can be attached to attributes, e.g., `>1600` for the `area` attribute. Joins can be expressed simply by linking the corresponding attributes. Here, we can relate home and crime data by joining on the region.

Once the condition of the query has been defined, the query output can be constructed. To this end, the user can drag subtrees from the condition window and drop them into the answer window. Additionally, the output can be restructured and new element names can be created (e.g., `in_region`). A special feature of BBQ is the `COLLECT` operator (Fig. 2, right), which allows to create data collections simply by clicking on the corresponding element subtree. In this way, all elements of that type which are retrieved by the query body will appear as a *collection list* in the answer. In Fig. 2, all `in_region` elements are collected as a list within the `answer` element. Within each `in_region` element, the `homes` subelement contains a collection of `home` elements. The crux of this feature is that it induces an *implicit grouping semantics*. In the above setting, homes are implicitly grouped by regions. Another feature are *ordering functions* that can be applied to collections: By default, elements of a collection are returned in the order they were found, but other element orders can be easily defined, e.g., based on attribute values or by applying list functions to them.

XMAS Queries. The output of BBQ is a XMAS query. In addition, since BBQ supports the generation of complex XML queries, it can also be used by the "mediation engineer" as a design tool for the mediator view. The syntax of XMAS queries resembles XML-QL. For example, for the user query above, the XMAS query in Fig. 3 is generated:

The query *body* (`WHERE`-clause) corresponds to the pattern constructed in BBQ's condition window; it defines conditions on the home and crime data in the mediator view. For example, the region `$R` of a given home `$H` must coincide with the name of the police service region of `crimes`, thereby defining a join condition. Observe that the variables `$H` and `$S` bind to elements, whereas all other variables bind to attribute values. The query *head* (`CONSTRUCT`-clause) specifies how the desired answer is structured. An expression like <in_region>...</in_region> {$R} defines a *collection* with *label* `{$R}` and specifies (i) that *for each* value of `$R`, *exactly one* `in_region` element is

and `www.sannet.gov`.

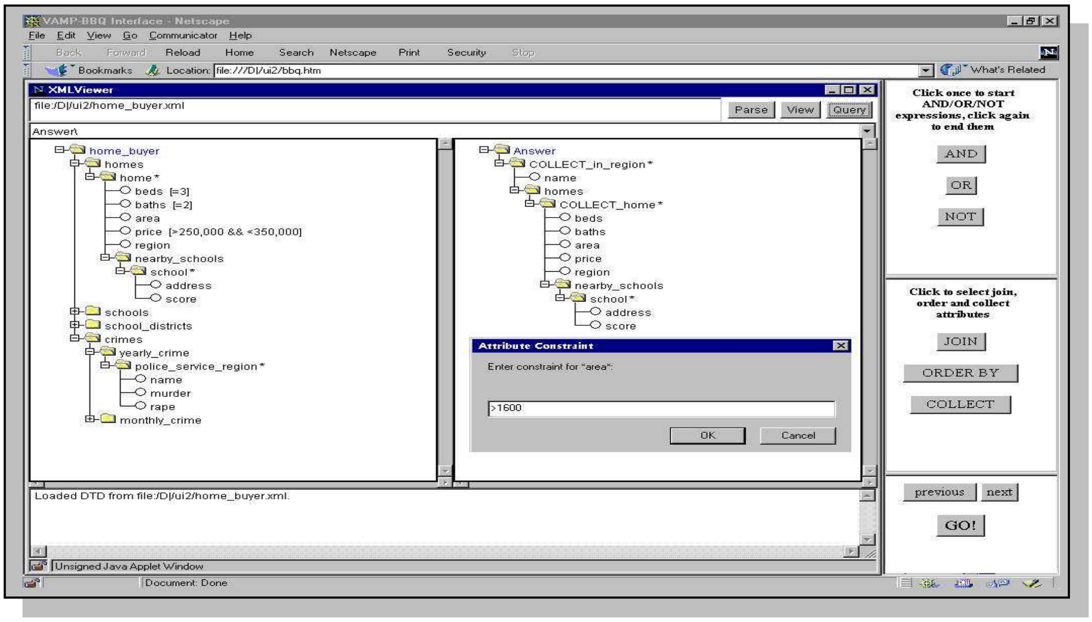

Figure 2: XML query composition with BBQ (left: *condition window*, middle: *answer window*, right: *operators*)

```
CONSTRUCT <answer> <in_region name=$R>
                      <homes>
                        ORDER_BY($H {$H}, price)
                      </homes>
                   </in_region> {$R}
          </answer>
WHERE <home_buyer>
          <homes> $H: <home region=$R beds=$BE
                       baths=$BA area=$A price=$P>
                         <nearby_schools>
                           $S: <school score=$SC/>
                         </nearby_schools>
                      </home>
          </homes>
          <crimes>    <yearly_crime>
                          <police_service_region name=$R
                           murder=$CM rape=$CRA/>
                      </yearly_crime>
          </crimes>
      </home_buyer> IN "www.sdsc.edu/MIX/med-view"
  AND   $BE=3 AND $BA=2 AND $A>1600 AND $P>250000
  AND   $P<350000 AND $SC>=70 AND $CM+$CRA=<15 .
```

Figure 3: A XMAS query

created, and (ii) that these elements become children of the surrounding parent element (here: `answer`). Given the fixed value for `$R` within each `in_region` element, a collection of homes is defined by `$H {$H}` (sorted by price). If no order function is given, a default order is used to determine the order within a collection. Instead of explicit collection labels, expressions of the form `[`*expr*`]` can be used. Roughly speaking, the collection label for `[`*expr*`]` comprises all free variables of *expr* and thereby induces an implicit *grouping* of data by those variables which do *not* occur in *expr*. Thus, for each binding of these "surrounding" group-by variables, exactly one *expr*-list is generated, which often yields the intended semantics and relieves the user from specifying labels explicitly.

4 Summary

MIX*m* is a fully XML-based mediator prototype whose graphical user interface BBQ blends browsing and querying of XML data. Powerful grouping and ordering operators are supported by means of novel collection list construct. BBQ automatically generates XML queries from the graphical query specification given by the user. The system solely relies on the XMAS query language for extracting XML data (both from the mediator and from the sources). An algorithm for DTD inference for certain views has been developed. In the demonstration, the system is not only presented from the end user's (BBQ) perspective, but also details of the generated XMAS queries and algebra expressions can be shown.

Acknowledgments. We thank Victor Vianu for enlightening discussions on the meaning of XMAS and Andreas Yannakopoulos for initial work on BBQ.