XML-Based Information Mediation with MIX

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

1 Background

As more and more data is made available through the Web, mediation of information from heterogeneous sources becomes a crucial task for future Web information systems. We describe the features of our information mediator Mix (***M**ediation of **I**nformation using **X**ML*), which is being developed as part of a joint project between SDSC and UCSD. Like its predecessor TSIMMIS [PAGM96], Mix relies on the well-known mediator architecture [Wie92] to provide the user with an integrated view of the underlying sources. To facilitate a uniform and flexible representation of arbitrary source data, Mix employs XML, which is not only a document mark-up language, but also a semistructured data model. However, the flexibility of the semistructured model can become its own stumbling block when schema information is "buried" within the actual data and the user is faced with the problem of formulating meaningful queries against the semistructured database.

As a solution to this problem, we use XML DTDs as a structural description (in effect, a "schema") of the data exchanged by the components of the mediator architecture. More precisely, we focus on *valid* XML documents, i.e., documents which conform to an associated DTD. The schema provided by a DTD is more versatile than relational schemas, and at the same time provides more structure than the plain semistructured model of existing approaches like TSIMMIS. Given the central role of DTDs in our approach, (semi-)automatic inference of view DTDs becomes an important issue. The DTD inference task is to infer the DTDs of the mediator view, given the mediator view definition and the source DTDs; see [PV99] for an algorithm on Mix's DTD inference.

The novel features of the Mix 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 a high-level, declarative query language Xmas[1], which builds upon ideas of languages like XML-QL, Yat, MSL, and UnQL [XML98, CDSS98, PAGM96, BDFS97]. For example, Xmas allows object fusion and pattern matching on the input XML data. 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 completely driven by the mediator view DTD and integrates browsing and querying of XML data. Complex queries can be constructed in an intuitive way, which resembles QBE. Due to the nested nature of XML data and DTDs, Bbq employs a novel graphical way to specify the nesting and grouping of query results.

In Section 2, we briefly discuss the overall architecture of the system. A closer look at the corresponding modules is given in Section 3 using a concrete example. Section 4 summarizes the main points of the prototype and its demonstration.

[1] ***XML M**atching **A**nd **S**tructuring Language*

2 MIX Architecture

The main architecture of MIX is depicted in Fig. 1: The *graphical user interface* BBQ allows the construction of queries in an intuitive way. BBQ is driven by the XML DTDs of the mediator view and guides the user in formulating complex queries. A design goal of BBQ is to provide a seamless blend of browsing and querying modes, in the spirit of GARLIC's PESTO interface [CHMW96].

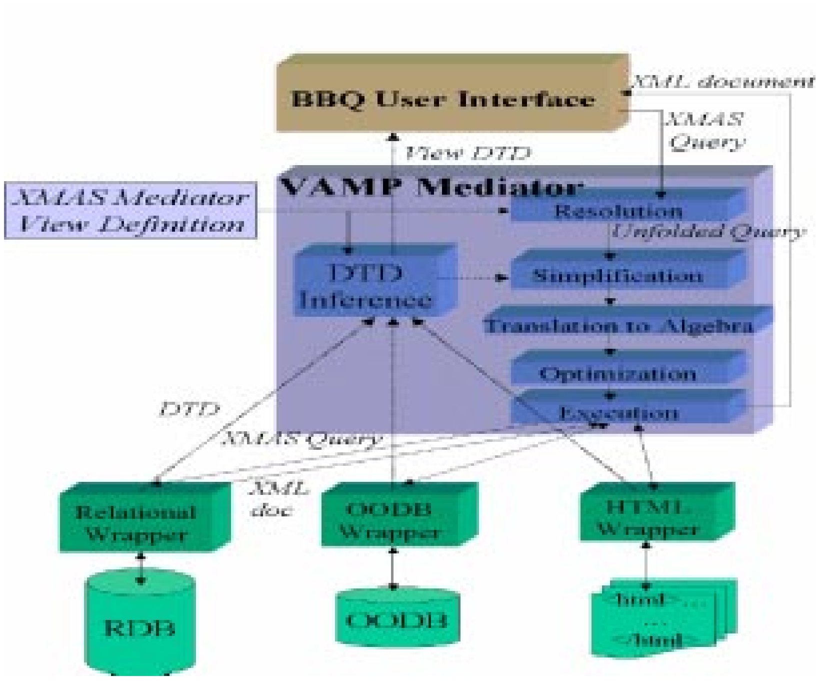

Figure 1: MIX architecture

The MIX *mediator* comprises several modules to accomplish the integration; its main inputs are XMAS queries generated by BBQ, and the *mediator view definition* (also in XMAS) for the integrated view. The latter has to be provided by the "mediation engineer", and prescribe how the integrated data combines the wrapper views. The *resolution* module resolves the user query with the mediator view definition, resulting in a set of unfolded XML queries that refer to the wrapper views. These queries can be further simplified based on the underlying XML DTDs.

The DTD inference module can be used to automatically derive view DTDs from source DTDs and view definitions, thereby supporting the integration task of the mediation engineer.[2] The *translation* module maps the simplified queries into the XMAS algebra which can then be further optimized. Finally, the execution engine issues XMAS queries against the wrappers, and returns the requested XML data to the user, after integrating the retrieved data according to the mediator view.

Wrappers are used on top of the heterogeneous sources to export data in a uniform format to the mediator. In contrast to previous approaches, which rely on "schema-less" semistructured models, we use XML as our model for semistructured data and exploit the structuring information provided by XML DTDs. Depending on the source's capabilities, a wrapper may support only certain types of XML queries, in which case it is the meditor's responsibility to issue only such queries.

Similar to TSIMMIS, MIX's query evaluation corresponds to a *lazy* approach (also called *on demand* or *virtual*), i.e., XML queries (expressed in XMAS) are unfolded and rewritten at runtime as they flow downwards from the user to the sources. In contrast, for example STRUDEL [FFK+97] and FLORID [LHL+98] follow the *eager* (or *warehousing*) approach, where data integration occurs in a separate materialization step, *before* the actual user queries.

[2]In MIX, DTD inference is performed in a separate off-line step, i.e., before the user interacts with the mediator.

3 Running Example

MIX is part of the DIVA project (***D**igital Government **I**nitiative for **V**irtual **A**gencies*) at SDSC/UCSD, whose goal is to develop tools and techniques for XML-based information mediation in the context of Government agencies. Clearly, MIX's general mediator architecture is not restricted to a specific application domain. Hence, we illustrate MIX's main features by means of a more common application, the *home buyer's scenario.*

Home Buyer's Scenario. Imagine a user who wants to buy a home in a certain region and who wants to make use of information available from the Web. A sophisticated query she may want to 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.[3] In our case, this involves sources HOME, SCHOOL, SCHOOL-DISTRICT, and CRIME.[4] Next, a "mediation engineer" integrates the sources by providing an appropriate *mediator view definition* in XMAS. Now the user can start to query the mediator through the BBQ user interface.

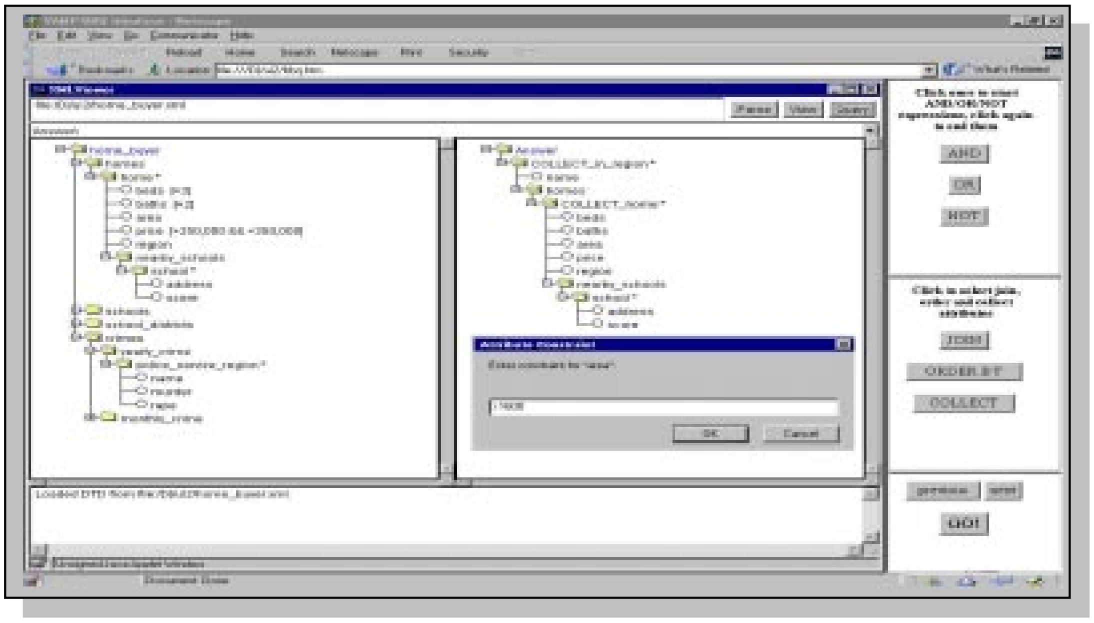

Figure 2: XML query composition with BBQ (left: *condition window*, middle: *answer window*, right: *operators*)

BBQ in a Nutshell. The graphical BBQ user interface 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 actual answers returned through the mediator are displayed in a separate window and can be browsed in the usual way. Since the retrieved answers are valid XML documents, they can be queried through BBQ as well. In this way, BBQ smoothly integrates browsing and querying of XML data.

Initially, the condition window shows the root element ⟨`home_buyer`⟩ of the mediator view. Now the user can successively click on subelements of ⟨`home_buyer`⟩ thereby exposing as much of the DTD structure as

[3]The need for such wrappers may decrease as more XML data becomes available in XML form.

[4]Their actual URLs are `www.realtor.com`, `www.ads.com`, `homeadvisor.msn.com`, and `www.sannet.gov`.

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. If the mediation engineer has included information about joinable attributes in the view DTD, these can bed highlighted by Bbq. Here, we can relate home and crime data by joining on the region, i.e., by linking ⟨`home_buyer`⟩.⟨`homes`⟩.⟨`home`⟩.⟨`region`⟩ with ⟨`home_buyer`⟩.⟨`crimes`⟩.⟨`yearly_crime`⟩.⟨`police_service_region`⟩.⟨`name`⟩.

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.* Roughly speaking, in the above setting, homes are implicitly grouped by regions, since ⟨`in_region`⟩ is a parent of ⟨`homes`⟩; see below for details. Another novel feature are versatile *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 borrows from XML-QL and is of the following form:

`CONSTRUCT` *head*
`WHERE` $body_1$ `IN` $source_1$
    (`AND`|`OR`) $body_2$ `IN` $source_2$
    ...
    (`AND`|`OR`) $body_n$ `IN` $source_n$
    (`AND`|`OR`) *predicate*

Here, $body_i$ is an XML template which is used to match and extract data found in $source_i$, and *predicate* specifies conditions on the variables occurring in $body_i$ ($i = 1, \ldots, n$). The variable bindings returned by the `WHERE`-clause are used to construct or refine an XML document. The *head* prescribes the details of this construction, most notably, grouping and ordering of elements. Typically, a mediator view definition will refer to several data sources by means of `IN`-clauses. In contrast, user queries generated by Bbq will only refer to one source, i.e., the mediator view.

For example, for our original user query, the following Xmas query is generated:

```
CONSTRUCT <Answer> [ <in_region name=$R>
                          <homes> [ $H ] ORDERBY $H.price  </homes>
                       </in_region> ]
          </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 .
```

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 whole elements, whereas all other variables bind to attribute values.

The query head (`CONSTRUCT`-clause) specifies how the desired answer is structured. Bracketed expressions denote *collection lists* and are the key concept for understanding XMAS ordering and grouping semantics. Here, the ⟨`Answer`⟩ element contains a list of ⟨`in_region`⟩ elements, each of which in turn contains a ⟨`home`⟩ collection which is ordered by the `price` attribute. XMAS allows to apply (built-in) ordering functions to act upon lists. If—like above—no ordering function is given, a default order is used (based on tree positions in the input, or the order in which data is found). A list expression `[`*expr*`]` in the head induces an implicit *grouping* of data. Roughly speaking, the group-by variables (in the SQL sense) are those variables in the head which do *not* occur in *expr*. Thus, for each binding of these "surrounding" group-by variables, exactly one *expr*-list is generated. In our example, this means that only one ⟨`Answer`⟩ element is created (since no head variables are outside the outermost list), and that *for each* region `$R`, only *one* ⟨`homes`⟩ element containing a *list* of ⟨`home`⟩ elements is defined. We believe that this grouping semantics is very natural and intuitive (e.g., it avoids the use of Skolem functions for grouping).

4 Demonstration Summary

MIX is a fully XML-based DTD-driven mediator prototype, which is developed in the DIVA collaboration between San Diego Supercomputer Center (SDSC) and the database group at CSE/UCSD. The main components of the prototype are operational and will be demonstrated at a local workshop with local government representatives in January 1999. MIX contains a versatile graphical user interface BBQ which blends browsing and querying of XML data and supports powerful grouping and ordering operators 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 automatic DTD inference 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 of the unfolding and rewriting steps within the mediator are shown.

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

References

[BDFS97] P. Buneman, S. B. Davidson, M. F. Fernandez, and D. Suciu. Adding Structure to Unstructured Data. In *6th Intl. Conference on Database Theory (ICDT)*, LNCS 1186, pp. 336–350, Delphi, Greece, 1997. Springer.

[CDSS98] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your Mediators Need Data Conversion! In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pp. 177–188, 1998.

[CHMW96] M. J. Carey, L. M. Haas, V. Maganty, and J. H. Williams. PESTO: An Integrated Query/Browser for Object Databases. In *Intl. Conference on Very Large Data Bases (VLDB)*, pp. 203–214, 1996.

[FFK+97] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. STRUDEL: A Web-site Management System. In *ACM Intl. Conference on Management of Data (SIGMOD)*, pp. 549–552, 1997.

[LHL+98] B. Ludäscher, R. Himmeröder, G. Lausen, W. May, and C. Schlepphorst. Managing Semistructured Data with FLORID: A Deductive Object-Oriented Perspective. *Information Systems*, 23(8), 1998. to appear.

[PAGM96] Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object Fusion in Mediator Systems. In *Intl. Conference on Very Large Data Bases (VLDB)*, 1996.

[PV99] Y. Papakonstantinou and P. Velikhov. Enhancing Semistructured Data Mediators with Document Type Definitions. In *Intl. Conference on Data Engineering (ICDE)*, Syndey, Australia, 1999. to appear.

[Wie92] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3):38–49, 1992.

[XML98] XML-QL: A Query Language for XML. W3C note, `http://www.w3.org/TR/NOTE-xml-ql`, 1998.