XML: Current Developments and Future Challenges for the Database Community

Stefano Ceri, Piero Fraternali, and Stefano Paraboschi

Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci, 32
Milano, Italy I-20133
ceri/fraterna/parabosc@elet.polimi.it

Abstract. While we can take as a fact "the Web changes everything", we argue that "XML is the means" for such a change to make a significant step forward. We therefore regard XML-related research as the most promising and challenging direction for the community of database researchers. In this paper, we approach XML-related research by taking three progressive perspectives. We first consider XML as a data representation standard (in the small), then as a data interchange standard (in the large), and finally as a basis for building a new repository technology. After a broad and necessarily coarse-grain analysis, we turn our focus to three specific research projects which are currently ongoing at the Politecnico di Milano, concerned with XML query languages, with active document management, and with XML-based specifications of Web sites.

1 Introduction

XML is currently in the process of replacing HTML as standard document markup language for the Web; it was developed as an evolution and simplification of SGML, another document markup language which had been available for more than a decade without really influencing the development of Web applications. Database researchers had very little impact on XML design, yet XML improves over HTML along two of the main directions of interest of our research community, i.e., providing data semantics and data independence.

- **Data semantics** is introduced within XML documents by means of *semantic tags* which annotate XML documents; we say that XML documents are self-describing, as the semantic annotations that each document carries along provide information about the document's content. In addition, XML documents may comply with a *document type definition* (DTD), a specification that is given separately from the document, a sort of document schema, that indicates the generic structure of one or more XML documents.
- **Data independence** is achieved because XML documents are specified independently from their presentation. Document presentation is the subject of a companion standard (XSL) which dictates how a *style sheet* can be

associated with an XML document in order to present it on a Web browser. Therefore, issues such as the choice of character sets used to represent a given portion of XML document, the size of these characters, or the various ways of emphasizing character strings are all omitted from the document specification in XML.

The strengths of XML in the domains of data semantics and data independence have slowly been understood and fully appreciated in all their positive implications; they are causing a progressive shift in the emphasis given to the XML standard. XML is turning from a pure document markup language into being considered as an extremely powerful data interchange format, i.e., an instrument for enabling data publication by various applications which need to co-operate. In essence, XML is now being considered as the key enabling concept for achieving data interoperability, a long-term objective of database research.

This shift in the emphasis on XML is becoming evident also within the W3C Consortium. In the "XML Initiative Web page" at http://w3c.org/XML we find the following description of XML goals. XML will:

- *Enable internationalized media-independent electronic publishing.*
- *Allow industries to define platform-independent protocols for the exchange of data, especially the data of electronic commerce.*
- *Deliver information to user agents in a form that allows automatic processing after receipt.*
- *Make it easy for people to process data using inexpensive software.*
- *Allow people to display information the way they want it.*
- *Provide metadata – data about information – that will help people find information and help information producers and consumers find each other.*

The above goals clearly position XML as vehicle for data exchange; they also emphasize the central role that is still played by data and data repositories in the deployment of applications, most notably for e-commerce, and the increasing importance of metadata in future Web applications. The database research community is thus strategically positioned "in the middle" of this evolution stream, and should not miss the many opportunities that are currently offered for driving the stream. In this paper, we analyze XML from three main perspectives, by looking at it first as a data representation standard (in the small), then as a data interchange standard (in the large) and finally as a basis for building a new repository technology. We next describe the recent research that has been carried out at our institute (Politecnico di Milano) on XML.

2 XML as a Data Representation Standard

We approach our survey of research directions on XML by first taking the viewpoint of XML as a data representation standard. In this context, we need abstractions for modeling, querying, viewing, updating, constraining, and mining a collection of XML documents. This viewpoint emphasizes the need for managing XML data as a single collection "in the small" (although such small world can be very large), disregarding for the time being data interchange and interoperability.

2.1 Data Modeling with XML

The notion of DTD, which can be associated with XML documents, introduces data modeling in the XML world. By means of DTDs, it is possible to specify a hierarchy of concepts (or elements) that constitute the XML document; each element may contain PCDATA (i.e., text strings that can be suitably parsed and encoded), attributes (i.e., properties given in the format of pairs <attribute_name, attribute_value>), and then recursively other elements, with arbitrary cardinality (at most one, exactly one, zero or more, one or more). One element may contain another element chosen among a list of several alternative elements. XML documents are lacking the notion of elementary types (such as integer or floating point numbers), as the only supported type in a document is that of PCDATA. For their ability of precisely describing recursive structures (with iterations, optionalities, alternatives, and so on), DTDs correspond to a grammar; an XML document is *valid* relative to its DTD if it can be produced by that grammar.

DTDs have several analogies with object-oriented data models. Every XML element can be considered as equivalent to an object, whereas the corresponding DTD element can be considered as equivalent to an object class. Each object can explicitly be associated with its object identifier (ID attribute), and objects can refer to other objects (IDREF and IDREFS attributes); however, IDREFs are not typed, hence references from one DTD element are not constrained to refer to instances of a given DTD element. The use of alternatives corresponds to union types, a feature that is rarely found in object-oriented models. The missing features with respect to object-oriented data models include class hierarchies, typed object references, and certain integrity constraints; these are the subject of an extension of XML, called XML Schema, which is currently under definition by the W3C (and addressed later on in the paper).

Due to this analogy, we could consider DTD design an instance of the generic problem of database design; indeed, we expect that this will sooner or later be recognized, and that therefore DTD design will be driven by data design abstractions and methods. There is a strong analogy between designing the DTD and conceptually designing a Web application modeled in XML, and Web modeling is in turn becoming a popular subject [7]. So far, however, this was rarely recognized, mainly because there are few instances of XML data and even fewer instances of top-down-designed XML data. Certain data design aspects cause conceptual difficulties and non-obvious trade-offs, such as:

- The alternative use of either attributes or element containment for modeling the same reality.
- How to deal with the substructuring of PCDATA contained by a given element (a typical feature of XML seen as a markup language but not obviously modeled and managed by XML repositories).
- How to deal with element ordering.
- In general, how to deal with all the missing integrity constraints.

Another interesting problem is that of **inferring a DTD** for XML data when this is not natively provided with a DTD. The inference of a DTD should be

driven by well-defined requirements, leading to the identification of synthetic DTD structures capturing most of the structure of the XML document, so as to further infer properties that may be useful for its efficient data storage and retrieval. A similar problem is inferring a common DTD for two distinct documents which are merged into a single one. Work in this direction was done at Stanford University in the context of Lorel by defining the notion of *data guides* [12].

2.2 Querying XML

Many efforts of the XML research community are focused on designing a standard XML query language. The first initiative in the field occurred when Dan Suciu and colleagues from AT&T and INRIA deposited a proposed XML query language, called XML-QL [9, 11], as a request for standardization to the W3C, thus fueling the debate on query languages for XML. Even before, the language Lorel had been defined at Stanford by Serge Abiteboul, Jennifer Widom and colleagues in the context of semi-structured databases and then found fully adequate for supporting XML queries [1]. The W3C has then taken the lead in driving the efforts towards a standard XML query language, first by organizing a well-attended workshop [17], next by starting a working group focused on the problem, which is expected to produce a standard recommendation by the end of the year 2000 (and hopefully sooner).

In a recent article [4] we carefully compared five XML query languages which are currently proposed. Besides the already mentioned XML-QL and Lorel, we reviewed also the query language facilities already present in XSL [19] and then XQL [15], a simple query language for selecting and filtering XML documents. Finally, we considered XML-GL, a language proposed within Politecnico offering a nice graph-based graphical interface [5]. In that article, we listed several features that should be possessed by an XQL query language, and then analyzed the available documentation (and in certain cases the prototypes) of the languages to check whether those features were supported.

The result of the analysis is represented in Figure 1, borrowed from [4]. The figure shows that languages can be broadly classified into two classes, the one of *expressive, multi-document query languages* and the one of *single-document query languages*; the main distinction between the two classes is the ability of joining two documents from arbitrary data sources. Between the two classes, XML-GL offers an easy-to-use paradigm, based on the fact that DTDs can be described as hierarchical structures (interconnected by references) and similarly queries can be represented as selections and annotations of those interconnected hierarchies. Although Lorel and XML-QL appear to be comparable in their expressive power (modulo some extensions required to XML-QL), the two languages present quite a different syntactic style; Lorel is deliberately very similar to OQL, while XML-QL is based on an XML-like hierarchical organization of tags, and thus appears much more verbose than Lorel. This can be best appreciated by looking at comparative query examples in [4].

In general, Figure 1 shows the need for:

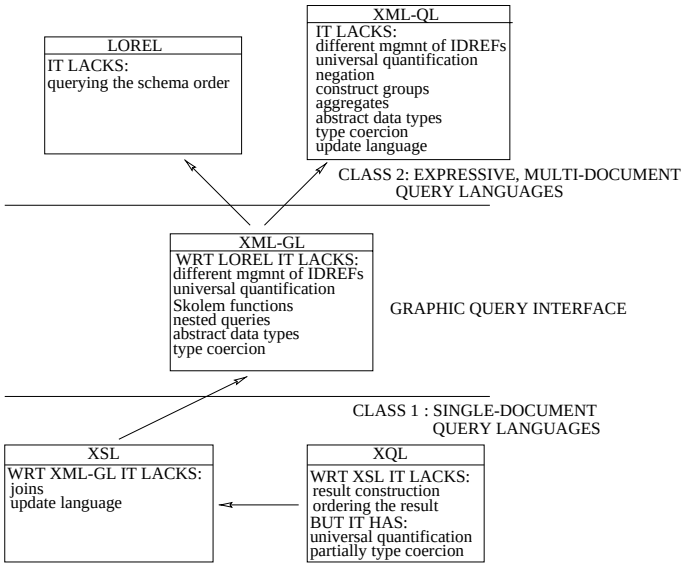

Fig. 1. Classification of query languages for XML (from [4])

- A powerful query language, covering all the aspects that are available in a typical SQL- or OQL-like language, normally used from within a suitable API.
- A sublanguage of the above, implementing simple selections and projections of XML documents, possibly compatible with XPath, the path expression language designed by the W3C XSL working group and already defined within the XSL standard.
- A graphic, QBE-like language allowing users to express a subset of the query language identified above in a simple, user-friendly way. Such graphic language should be offered within an interface capable of alternating between the visual and textual expression of queries, as it is customary in many database products.

The field of query language design appears to be frozen until the W3C Working Group will present its results, but a lot of language-independent research can still take place along several dimensions, including classical ones such as:

- Defining an **XML algebra**, i.e., defining a suite of orthogonal and minimal algebraic operators that procedurally define a strategy for expressing a given query. Requirements of the query language on the algebra can be abstracted by looking at the expressive power of Lorel, the most expressive query language, or else by looking to the collection of orthogonal features listed in [4]. Preliminary definitions of XML algebras are given in [10, 2].
- Based upon the algebra, studying the **equivalence of algebraic expressions** (for better understanding the expressive power of XML queries) and

then studying, among equivalence transformations, those giving an **optimization of algebraic expressions**, in the same exact way as it is done with relational databases.
- Defining as well an **XML calculus** and XML logic formalism that may give hints upon how to achieve more declarative languages, possibly empowered with recursion and negation.

Besides these classical - and to some extent foreseeable - research dimensions, other research topics are more concerned with the use of XML query languages for searching the Web. We mention:

- **Proximity search**, i.e., search for concepts which are related to given concepts due to their vicinity; this, in turn, can be measured conceptually (e.g., based on the deep meaning) or physically (e.g., based on their position within a hypertext).
- Giving **approximate results**, i.e., results that may be sufficient to characterize the query answer and whose computation can take place more efficiently. In this context, it is important to define ranking schemes and also usage patterns for progressively refining query results based on the ranking.
- Combining a full-fledged XML query with **keyword-based search**, which has proven to be enormously efficient when employed from within the Web search engines.

2.3 Beyond the XML Query Language

An XML query language is an essential ingredient at this stage of XML development, but its definition will open up new problems and issues. We survey some of them.

- A query language immediately brings about the notion of **view**. XML views will be particularly relevant as one can envision that XML-based sites could be views of other XML-based sites, thereby building hierarchies of derived XML data. Such an architecture is already very popular in the context of data warehousing. Views bring with them the issues of their materialization and incremental maintenance, which are classical problems of replication management.
- It is essential to define **semantic constraints** (e.g., referential integrity and more general path constraints) and then make use of them in order to filter illegal XML documents or to optimize query processing.
- Similarly, it is essential to extend the query language with an **update language**, capable of systematically applying changes to selected portions of XML documents.
- Along this direction, once updates become supported, then it becomes possible to support active rules or **triggers**, capable of performing reactive processing upon XML documents. Later on, in Section 5.2, we briefly describe the features of Active XML-GL, a trigger language which extends XML-GL.
- Finally, we expect that XML-based data collections will be ideal for supporting novel approaches to **data mining**, where the existence of semantic tagging may help in structuring the knowledge discovery process.

3 XML as a Data Interchange Standard

Data interchange standards are based on the use of SQL, called "intergalactic dataspeak" by Mike Stonebraker, and typically embedded within standard APIs, such as ODBC and JDBC. In the object-oriented world, CORBA and DCOM provide location-transparent method invocation. All of these standards, however, operate by invoking functions on (or shipping functions to) remote data stores; instead, XML promises to enable data interchange, thereby rising the level of interoperability.

The success of XML as a data interchange standard depends largely from the fact that XML is self-describing, due to semantic tags. Therefore, the publisher of XML data provides also the relative meta-information, thereby enabling the correct interpretation of the XML data by the client. When XML data is extracted from relational databases, metadata is simply a well-defined DTD derived from the relational schema, to which the XML data must rigidly comply. However, XML data may also be extracted from arbitrary legacy systems by a wrapper (which should add to XML data suitable meta-information) or may describe arbitrary semi-structured documents.

Assuming that each data provider publishes XML content, it then becomes possible to pursue three of the W3C goals as defined in the introduction section: define platform-independent protocols for the exchange of data, especially the data of electronic commerce; deliver information to user agents in a form that allows automatic processing after receipt; and provide metadata that will help people find information and help information producers and consumers find each other. The above goals open up a number of research problems, including:

- The development of **wrapping technology** helping in the semi-automatic publishing of legacy data in XML.
- The establishing of well-understood **E-commerce protocols** enabling negotiation and bidding in the context of many-to-many buyers and sellers.
- The use of **agent technology** for automatic discovery of information and negotiation in behalf of clients (e.g., searching for the best offer in the Web shops).
- The development of a new generation of XML-based **search engines** capable of extracting from the Web those specific portions of XML documents dealing with given semantic properties.

The skeptical observer may note that "XML is just syntax", and therefore it does not solve the problems of semantic interoperability. However, a variety of XML-based semantic descriptions are being defined for specific domains. The site http://www.xml.org currently lists several proposals for different applicative domains (e.g., for genetic data [16], mathematical data [20], chemical data [8]). By adopting a domain-specific tag encoding, data providers are guaranteed to share a data interpretation consistent with their customers.

We expect that every scientific community will understand the importance of data interchange through XML and then develop its own XML-based ontology. Specifically, the community of computer scientists and engineers will specialize

in the **modeling of computer systems specifications**; these are typically developed within co-operative and distributed contexts and require to be stored in common repositories, hence will take strong advantage from being collected in a format that warrants effective data interchange. The XMI standard [14] is already pursuing this direction; we will describe in Section 5.3 a new specification language for Web applications, called WebML, which is based on XML.

3.1 Beyond XML

DTDs of XML provide a rich collection of structuring primitives, but are clearly incomplete with respect to the data definition primitives available for database schemas, and therefore carry less semantics. This issue is being covered by a W3C working group defining XML Schema, which aims at extending DTDs in the following dimensions:

- Support of (basic) **data types**, thereby adding classical types (such as integers, floating point numbers, or dates) to text (PCDATA) currently supported in XML. Data types in XML Schema are extensible, in the sense that they are autonomously defined and as such can be redefined or augmented without requiring the redefinition of the standard.
- Support of **typed links**, i.e., of links connecting instances of a given element to instances of another, given element.
- Support of **integrity constraints**, including multi-attribute primary and foreign keys, minimum and maximum cardinality constraints, domain and range constraints.

A full description of XML Schema is outside of the scope of this paper and can be found in [22, 23]. We observe that XML Schema is very much complete and coherent with respect to the canonical approach to data design and as such it may be too demanding to become really widespread; however, it may serve the need of integrating, within an XML-compliant formalism, the information which is normally available in the data dictionary of many DBMSs.

4 Repository Technology for XML

XML would not be so popular if it were not already backed by a first generation of XML repository systems, generically denoted as **XML servers**.The success of these systems is due in great part to the *Document Object Model* [18], which offers to applications a portable interface for access and management of XML objects. XML servers provide to researchers concrete evidence that XML technology has all the potential to be well supported, and actually to scale to very large sizes. Several ongoing projects are focused on building repositories for hosting native XML data, as opposed to hosting data in a different format and then wrapping tags "around" it.

Current XML server technology aims at offering services for inserting and retrieving XML objects, normally stored as flat files, using a more efficient format, e.g., pure or interconnected trees. The big debate currently ongoing among

database researchers is focused on whether the representation of XML data should be flat and DTD-independent or instead should be more complex and influenced by the DTD (or by the XML Schema) of the stored document. The former solution is clearly indicated when XML data is unstructured, because the mapping of each element to a table would lead to sparse and inefficient databases; the latter solution could be beneficial with structured XML data, and in perspective could guarantee better query processing performance (i.e., better support of the standard XML query language).

Although mapping each element to a table seems too inefficient, we expect that in the long run the winning solution will be found by adapting relational storage servers to the need of XML data, in the same way as many relational storage servers currently support object-oriented and multi-media data management. We also expect that the flat vs structured modeling dilemma will be best solved by hybrid storage structures, and that this will lead to **flexible storage technology**, i.e., one where the physical data representation will be tailored to the kind of XML data to be stored. This will introduce the need not only of providing the underlying storage technology, but also of providing suitable **data storage methods**, that should infer the best physical mapping based on the features of XML data and then be able to reorganize such mapping dynamically, for instance in order to support complex queries.

We also believe that XML storage technology will naturally be distributed, in the sense that documents conceptually belonging to the same document class (and possibly described by the same DTD) will be hosted on several interconnected storage systems; therefore, we anticipate work in the direction of building **distributed XML storage systems**, possibly with data fragmentation and replication. The current efforts on the standardization of XML fragments (see [21]) may provide useful concepts for supporting a logical view of a document collection, irrespective of its physical fragmentation across many sites.

Optimizing the database engine is the current main focus of the database research community; this was considered as short-sighted by the VLDB Endowment, which recently issued a message to DBWorld asking for a redirection of research, emphasizing the need to reach out from the core database technology. We basically agree with such message, but we also note that query optimization for XML data will give to core database research a lot of challenges. Among them, we list new indexing techniques (e.g., supporting queries across links and along containment paths); how to exploit DTD knowledge or semantic constraints in order to speed up the query evaluation; how to deal with replicas and order; and so on. Going outside of a single repository, distributed query processing will be exacerbated by the fact that data sources are potentially very many and not known in advance; therefore, query processing strategies will probably be adaptive. Finally, performances will be achieved by means of parallelism, which may be either at the strategy level (by spawning multiple concurrent searches over distributed XML data) or at the physical level (by XML repositories hosted on multi-processor platforms).

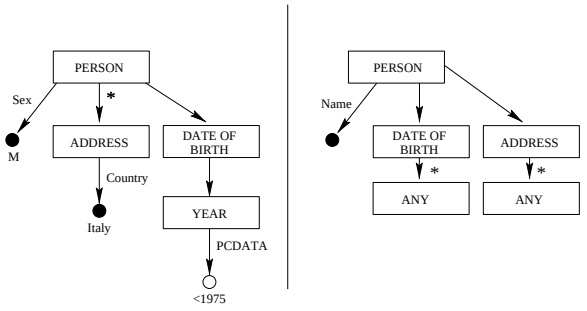

Fig. 2. An XML-GL query

We agree with Alon Levy [13] that we need to reconsider the measuring of performance, and that the new measures should take into account, among other factors, the irregularity of data and the number and heterogeneity of sources from which data are fetched.

5 Research Directions at Politecnico di Milano

After a broad and necessarily coarse-grain inspection of the very many directions of research for XML, we turn our focus to three specific research projects which are currently ongoing at the Politecnico di Milano.

5.1 XML-GL: Graphic Query Language for XML

An XML-GL query can be applied to an arbitrary XML document and produces a new XML document as result. The input document may or may not have a DTD: in the former case, the graphical representation of the DTD is the startpoint for the graphical expression of the query.

A basic XML-GL query is composed by a pair of graphs, called LHS and RHS graphs (see Figure 2, extracting all male residents of Milano born before 1975). Each graph is composed of labeled nodes (represented by rectangles or small circles) and by directed arcs connecting them. The rectangles correspond to elements, the circles to attributes and to terminal elements (i.e., elements which do not contain other elements). Unlabeled arcs represent element containment; labeled arcs represent references among elements, (IDREF attributes). The `*` (star) operator on arcs represents an arbitrary navigation along the arcs, and the `any` node matches any element in the document.

The LHS graph identifies the information of interest in the document, specifying where the information must be found and which are the conditions on the data that must be satisfied. In the query in Figure 2, the LHS searches in the document the `PERSON` elements which contain: (1) an attribute `Sex` with value "M"; (2) an `ADDRESS` element, at an arbitrary level of containment below `PERSON`, containing an attribute `Country` with value "Italy"; and (3) a `DATE OF`

BIRTH element containing a YEAR element before 1975. The evaluation of the LHS produces a set of subgraphs, composed by the fragments of the XML document that can be successfully substituted to the graph in the LHS.

The RHS graph is responsible of the construction of the query result; each subgraph produced by the evaluation of the LHS of the query on the input document is elaborated according to the nodes in the RHS to generate a subgraph of the result. Nodes of the RHS and LHS are put in correspondence either by using the same node name, or (in presence of ambiguity) by means of unlabeled non-directed edges connecting one node in the LHS to exactly one node in the RHS. From a document processing point of view, the semantics of the RHS of XML-GL queries is similar to that of a *transformation program* that converts a tagged document into another one by means of pattern matching and rewriting, as proposed for instance in the XSL language [19]. In the example, the result of the query contains the PERSON elements that have been selected in the LHS, complete with attribute Name and their complete subelements DATE OF BIRTH and ADDRESS.

XML-GL offers a rich set of additional features, like unnesting and nesting of XML objects, element ordering, data sorting, arithmetic functions, and aggregate functions. The RHS graph may include nodes that permit a complete restructuring of the document, possibly obtained by combining several distinct documents. Overall, this research effort demonstrates that a graph-based approach is quite natural for XML documents and can offer an intuitive way to represent operations on them.

We are currently working on a tool for query expression which is capable of mapping XML-GL into a well-defined subset of Lorel and vice-versa, linked to the Lorel prototype [1] to support query execution. The objective of this effort is to prove the effectiveness of graphic formalisms for querying XML documents.

5.2 Active XML-GL: Active Rule Language for XML

As a follow-up of our effort on XML-GL, we defined an active rule language, called Active XML-GL. Rules are managed by an **active rule engine**, which operates in the context of an XML document server and may perform several tasks, such as the automatic synthesis of documents, or the checking and repair of integrity constraints, or the incremental maintenance and refresh of views and of related documents, or the implementation of push technology (i.e., the notification of document changes to selected users).

An example of an Active XML-GL rule is in Figure 3. The rule inserts into a personal address book the list of persons born in Italy before 1975. Each Active XML-GL rule follows the ECA (event-condition-action) paradigm, where:

- **Events** are changes to a document. Events may be detected continuously or they can be perceived when the document returns to the repository after having been processed by an application.
- **Conditions** are queries on the document base, expressed using the XML-GL query language; the query normally inspects the content of the changed document and possibly compares it with the content of a document base.

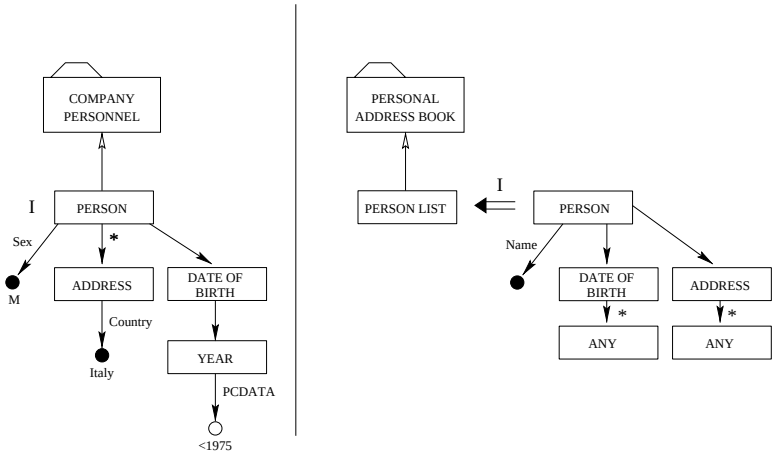

Fig. 3. An Active XML-GL rule

– **Actions** consist of building new documents and/or modifying existing documents in the document base, and then placing them into suitable folders, publishing them on the Web, or sending them by e-mail.

In order to express the above components, we extended the expressive power of XML-GL by adding:

– The representation of **atomic events** applicable to an XML document chosen as the *target* of the rule, with a clear semantics that enables to identify, when an event has occurred, which elements of a target document are affected by (bound to) the event. Events are specified as suitable labelings of the XML-GL graphs; the rule in Figure 3 reacts to insertions of `PERSON` elements. We also indicate the activity (i.e., document editing, document publishing in the Web, or message exchange) which causes the event occurrence; the event of the rule in the example occurs when a new element is inserted into the COMPANY PERSONNEL folder.
– The representation of **updates to XML documents**, enabling the declarative specification of the changes of content of XML documents, typically based upon the modified elements of the target document. Updates are defined graphically in the style of XML-GL. Also in this case, the graphical definition of the updates is easy to understand and offers a considerable simplification with respect to a textual description. In the example in Figure 3, the new `PERSON` element is added as a subelement to the `PERSON LIST` element in the PERSONAL ADDRESS BOOK folder.

Two alternatives set-oriented and instance-oriented semantics are supported; they correspond to the alternatives that are offered in relational systems (as introduced by `for each row` and `for each statement` clauses). In order to guarantee portability and independence of the rule engine, we do not assume that

change detection should necessarily be performed by specific, low-level changes on XML servers. Rather, we assume that documents can be arbitrarily checked-out for editing and then checked-in; similarly, they can be arbitrarily published on the Web, or received within an arbitrary message. Specific algorithms, called **XML-diff**, compute the difference between the initial and final XML document, expressed as a sequence of elementary changes (called *edit script*). The XML-diff algorithms use cost functions to identify, among all the possible edit scripts, those that minimize the editing costs. We are currently studying an important property of active rules, called *edit-script independence*, that holds when the rule semantics is not affected by the particular choice of edit script performed by an XML-diff algorithm. We are also studying the properties of rule sets execution in the presence of cascading rules and conflicts, as well as the properties of termination, confluence, and observable determinism.

5.3 WebML: XML-Based Web Site Specification Language

Web Modeling Language (WebML) is a specification language for Web applications that enables designers to express the core features of a site at a high level, without committing to implementation-specific details. WebML is based on an XML syntax, which can be processed by software generators for automatically producing the implementation of a Web site in specific mark-up and server-side scripting languages (e.g., HTML and Visual Basic Script). WebML concepts are also associated with an intuitive graphic representation, which can be supported by CASE tools and effectively communicated to non-technical site developers (see http://webml.org). WebML concepts are organized in four orthogonal models, called structural, hypertext, presentation, and personalization models.

The **structural model** expresses the data content of the site, in terms of the relevant entities and relationships; it is compatible with the E/R model, the ODMG object-oriented model, and UML class diagrams.

The **hypertext model** describes one or more hypertexts, which can be defined to publish the information of the structure schema in the site. The hypertext model is based on the following abstractions:

- A collection of six **unit types** (data, multidata, index, filter, scroller, and direct), which represent alternative ways of "publishing" objects or sets of objects defined in the structure schema.
- A **composition model**, which permits one to put units into pages and includes structuring mechanisms enabling to recursively compose pages into pages so as to define loadable content packages (e.g., frames in HTML).
- A **navigation model** for interconnecting pages into a hypertext. Links are either non-contextual, when they connect pages, or contextual, when they connect units.

The **presentation model** expresses the layout and graphic appearance of pages, independently of the output device and of the rendition language, by

means of an abstract XML syntax. Presentation specifications are either page-specific or generic; in the latter case, they are based on predefined models independent of the specific content of the page.

The **personalization model** is used to describe the delivery of personalized content, navigation, and presentation. Users and user groups are explicitly modeled in the structure schema in the form of predefined entities called User and Group. Then, OQL-like declarative expressions and ECA business rules can be added, which define derived content based on the profile data stored in the User and Group entities or reactions to events for updating user profile data [6].

By combining the WebML concepts of pages, units and links in different ways, it is possible to model arbitrarily complex Web sites. WebML specifications are the input of a novel Web design tool suite (called ToriiSoft, see *http://www.toriisoft.com*), which transforms high-level specifications into multiple output languages (such as HTML and WML), thus supporting multi-device output generation, and then provides the binding of templates to data using multiple server-side script languages (such as Microsoft's Active Server Pages and JavaSoft's Java Server Pages). WebML specifications are stored in an XML repository available to all the tools. The use of XML is fundamental to guarantee portability of the specifications and independence from output languages and template generators.

6 Conclusions

In this paper, we have given a broad overview of the research directions related to the XML standard. After the theorem that "the Web changes everything" [3] we believe in the corollary that "XML is the means" for such a change to make a big step forward. However, the future is always unpredictable, and so the forthcoming years will tell whether this corollary is true (the theorem is obviously true). We have traced a number of research directions, and we believe that the database research community should actively pursue all of them. Most of this research is short-to-medium term, but also XML penetration will either occur in the next few years or else be dropped in view of other priorities.

This road contains a number of new and challenging, "fully original" research topics; but it also contains a number of suggestions for consolidation, i.e., for bringing to the XML world a body of knowledge that the database research community has established over decades, working on relational and object-oriented technology. We believe that such a consolidation is important, as it will give a field-tested proof that a solid transfer of concepts can take place.

References

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1), Apr. 1997.

[2] D. Beech, A. Malhotra, and M. Rys. A formal data model and algebra for XML, 1999. Unpublished report.

[3] P. Bernstein, M. Brodie, S. Ceri, D. DeWitt, M. Franklin, H. Garcia-Molina, J. Gray, J. Held, J. Hellerstein, H. V. Jagadish, M. Lesk, D. Maier, J. Naughton, H. Pirahesh, M. Stonebraker, and J. Ullman. The Asilomar report on database research. *ACM SIGMOD Record*, 27(4):74–80, Dec. 1998.

[4] A. Bonifati and S. Ceri. Comparative analysis of five XML query languages. *ACM SIGMOD Record*, 2000. to appear.

[5] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A graphical language for querying and restructuring XML documents. In *Proc. of the 8th Int. World Wide Web Conf.*, Toronto, Canada, May 1999.

[6] S. Ceri, P. Fraternali, and S. Paraboschi. Data-driven, one-to-one web site generation for data-intensive applications. In *Proc. of 25th Conf. on Very Large Data Bases*, pages 615–626, Edinburgh, Scotland, Sept. 1999.

[7] P. P. Chen, D. W. Embley, and S. W. Liddle, editors. *Workshop on the World-Wide Web and Conceptual Modeling (WWWCM'99)*, Paris, France, Nov. 1999.

[8] Chemical Markup Language (CML), 1999. http://www.xml-cml.org/.

[9] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A query language for XML. In *QL'98 - W3C Workshop on Query Languages* [17].

[10] P. Fankhauser, M. Friedrich, G. Huck, I. Macherius, and J. Robie. Querying XML with locator semantics. Submitted for publication, 1999.

[11] M. Fernandez, D. Suciu, and A. Deutsch. XML-QL demo site, 1999. http://www.research.att.com/~mff/xmlql-demo/html/.

[12] R. Goldman and J. Widom. DataGuides: Enabling query formulation and optimization in semistructured databases. In *Proc. of the 23rd Int. Conf. on Very Large Data Bases*, pages 436–445, Athens, Greece, Aug. 1997.

[13] A. Levy. http://www.cs.washington.edu/homes/alon/widom-response.html. More on data management for XML, 1999.

[14] Object Management Group. XML Metadata Interchange (XMI), 1998. http://www.omg.org/.

[15] J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). In *Query Languages 98* [17].

[16] Visual Genomics, Inc. Bioinformatic Sequence Markup Language (BSML), 1999. http://visualgenomics.com/bsml/index.html.

[17] World Wide Web Consortium. *QL'98 - W3C Workshop on Query Languages*, Cambridge, Mass., Dec. 1998. http://www.w3.org/TandS/QL/QL98.

[18] World Wide Web Consortium. The Document Object Model (DOM) Level 1 Specification, October 1998. http://www.w3.org/TR/REC-DOM-Level-1/.

[19] World Wide Web Consortium. Extensible Stylesheet Language (XSL) Specification, 1999. http://www.w3.org/TR/WD-xsl.

[20] World Wide Web Consortium. Mathematical Markup Language (MathML) 1.01 specification, July 1999. http://www.w3.org/TR/REC-MathML.

[21] World Wide Web Consortium. XML Fragment Interchange, June 1999. http://www.w3.org/TR/WD-xml-fragment.

[22] World Wide Web Consortium. XML Schema part 1: Structures, Dec. 1999. http://www.w3.org/TR/xmlschema-1.

[23] World Wide Web Consortium. XML Schema part 2: Datatypes, Dec. 1999. http://www.w3.org/TR/xmlschema-2.