XML in Enterprise Systems: its Roles and Benefits

Jaroslav Pokorný

Charles University, Faculty of Mathematics and Physics,
Malostranske nam. 25, 118 00 Praha 1, Czech Republic
pokorny@ksi.mff.cuni.cz

Abstract. Enterprise information integration (EII) requires an accurate, precise and complete understanding of the disparate data sources, the needs of the information consumers, and how these map to the business concepts of the enterprise. In practice, such integration takes place in context of any enterprise information system. In the paper we explain various approaches to EII, its architectures as well as its association to enterprise application integration. We justify why XML technology contributes to finding sufficiently powerful support for EII. We present some features of the XML technology, mainly its database part, and show how it is usable in EII.

Keywords: XML, enterprise information integration, XQuery, XSLT, Web services, XML databases

1 Introduction

The language XML originally designed as a standard protocol for data exchange, serves today as a data model and background for databases of XML documents. Its main advantage is that it enables to create a background for applications beyond conventional data models, i.e. everywhere where we need, e.g., hierarchical data structures, recursive data structures, regular expressions, missing and/or duplicate data, and other non-traditional data requirements. XML creates a technological platform for Semantic Web.

A collection of languages, techniques, and standards developed by the World Wide Web Consortium (W3C[1]), called *XML technology* today, contributes to many application areas, as, e.g., B2B interactions, Web services, as well as, in general, to improvement of inter- and intra-enterprise applications. In the paper, we focus just on use of XML technology in enterprises.

Often an *enterprise information system* (EIS) is characterized as an information and reporting tool for the preparation, visualization, and analysis of operational enterprise data. An associated collection of activities and software components supporting accessing data from any source systems is then called *enterprise information integration* (EII). In other words, EII provides programmers with a single-site image of disparate data that may be maintained in different formats,

[1] http://www.w3.org/standards/xml/

retrieved via different application programming interfaces (APIs), and managed by different remote servers. The analyst community and other observers talk often about "virtual data federation". Data integration is crucial in large enterprises that own a multitude of data sources, like relational databases, Web services, files, and packaged applications. The same holds for offering good search capabilities across amounts of data sources on the Web.

Integration-related area contains also *enterprise application integration* (EAI). EAI integrates application systems by allowing them to communicate and exchange business transactions, messages, and data with each other using standard interfaces. It enables applications to access data transparently without knowing its location or format. EAI is usually employed for real-time operational business transaction processing. It supports a data propagation approach to data integration. A strong separation of EII and EAI can mean that enterprise data is accessed by an EII tool and updated by an EAI tool. EII solutions today should address both application and information integration.

A special form of data integration is required by data warehouses and business intelligence. *Extract*, *transform* and *load* (ETL) processes were considered the most effective way to load information into a data warehouse.

In general, EII needs [6] to

- support all information types, structured, unstructured and semi-structured,
- provide for context, i.e., where does the information fit in the schema or ontology of the receiving repository/application, and what are the relevant behavioural constraints.

Many enterprises today are moving towards the adoption of *service-oriented architectures* (SOAs) based on XML and Web services [5]. Web services represent a less costly and loosely-coupled approach for EII. Often Web services are considered as part of the EII whole. Consequently, a *service composition* gains strength in EIS today. A relatively little work has been done to facilitate integration at the *presentation* level, i.e. the development of user interfaces. This part of application development in EIS belongs to the most time-consuming activities. Other direction of the EAI industry is toward the use of an *enterprise service bus* (ESB) that supports the interconnection of legacy and packaged applications, and also Web services.

XML, enabling to declare and enforce structure of content, plays an important role both in EIS development and EII processes. The reason is simple. In the past, any exchanging information between content repositories and data-oriented applications within and across enterprise was very difficult due to incompatibility of supporting systems. Even data warehouse solutions were considered inappropriate for supporting such needs. EII vision is namely to provide tools for integrating data from multiple sources without first loading their data into a central warehouse. EII should perform the integration in real time on an on-demand basis. Emergence of XML made it possible to build EII on an XML data model and query language XQuery, i.e. with XQuery interface to these multiple sources.

The purpose of this work is to summarize some parts of the XML technology relevant for EII and show, how XML databases can help to create more responsive EII architectures. The remainder of the paper is organized as follows: Section 2 mentions some approaches to EII as well as commercial products based on these

approaches. After summarizing some basics of XML database technology in Section 3, in Section 4 we focus on applications of XML databases in EII. We mention also the concept of content management system there. Finally, Section 5 concludes and lists future work.

2 EII: Approaches and Related Works

By [18] EII is defined as the integration of data from multiple systems into a unified, consistent and accurate representation geared toward the viewing and manipulation of the data. Data is aggregated, restructured and relabelled (if necessary) and presented to the user. In viewing EII from a software engineering point of view, it is a type of middleware that allows companies to combine data from disparate sources into a single application.

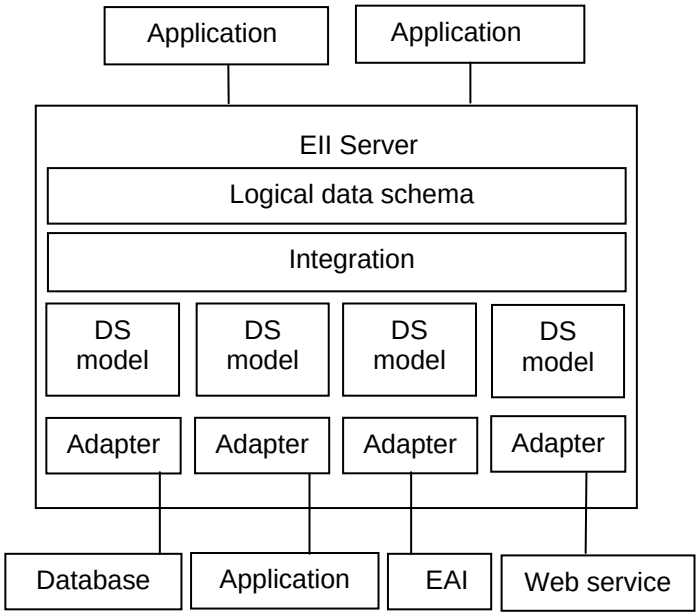

Fig. 1. EII approach to data integration

EII is based on a more flexible form of integration than simple data integration. EII data sources (DS) are viewed by applications as a single virtual database (see Figure 1). EII is based on a framework that exposes rather declarative interface for specification of integration requirements. EII provides applications a single, *virtual view* across multiple data sources. Applications access data sources through these views and through only one API of the EII server. Queries are transformed into queries against data sources. In other terminology, this approach is based on *mediation*. There is also a variant called *federation*, which integrates data by defining mappings between all pairs of schemas of the member databases. This variant called also a *loosely coupled federated system* is not broadly applied in real enterprise

environment because of the using of private protocol and data model, low performance, laborious process, critical implementation conditions, immature technology and the lack of reliable infrastructure [23]. Although the federated systems are relatively easy to implement, they do not scale well. By [10] an information integration infrastructure should support placing and managing data at multiple points in the data hierarchy to improve performance and availability. By the way, most of today's EII systems are really federated information systems.

To implement mediation, EII requires an accurate, precise and complete understanding of the disparate data sources, the needs of the information consumers, and how data model is mapped into a single, generic representation – a *logical data schema* that specifies the virtual view.

Therefore, the available approaches to EII can be considered based on the underlying logical model, the data transformation framework, and the query interface. Due to the well-known restrictions of relational data model in context of enterprise variety of data, it is not too perspective solution now. For example, iWay Data Hub[2] enables to create reusable relational views.

Purely XML-oriented approaches use XML as the logical data model. All data sources are represented as XML document collection and XQuery serves as the language of transformation as well as the query language. The EII server is a virtual XML database. As examples in this category we can mention Ipedo's XIP[3] and Liquid Data for WebLogic [3]. In XIP it is possible to query not only collections of XML documents, which is the best known use of XQuery, but also relational databases, web services, common data formats like CSV and fixed length formats. In addition, the Ipedo XQuery engine also allows users to create custom data sources and make them available to XQuery developers. In combination with the distributed SQL query engine, also offered in XIP, these capabilities represent one of the most powerful ways for EII.

However, a use of XML can be only a part of EII solution. There are tools, e.g., MetaMatrix [7], providing an integrated environment for modelling different types of data and information systems. In MetaMatrix different layers of metadata are created in more domain-specific languages (including XML Schema), i.e. XML is not a target language here. The support for multiple metamodels is ensured by OMG's MOF (Metadata Object Facility) architecture, i.e. relationships among metadata of different layers are expressed by mapping specifying transformations.

Although all the approaches have their advantages and disadvantages, the XML approach is excellent in data modelling and query capabilities, in particular with applications that use data from non-relational data sources, such as message queues, EJBs, XML documents, and Web services.

A more advanced approach to integration in EIS is enterprise mashups. Remind that a *mashup* is a Web application that combines content from two or more applications to create a new application. The applications can be built on-the-fly to solve a specific business problem. For example, Damia [1] is inspired by the Web 2.0 mashup phenomenon. It consists of (1) a browser-based user-interface that allows for the specification of data mashups as data flow graphs using a set of operators, (2) a

[2] http://www.iwaysoftware.com/products/eii.html
[3] http://www.ipedo.com/html/ipedo_xip.html

server with an execution engine, as well as (3) APIs for searching, debugging, executing and managing mashups.

Web mashups perform integration both at the application level and at the presentation level. Unfortunately, due to very little support both in terms of model and tools, the presentation part of mashups is developed manually today. An interesting approach to component integration at the presentation level is proposed in [20].

However, the mentioned approaches did nothing to address the semantic integration issues – sources can still share XML files whose tags are completely meaningless outside the application. In consequence, almost all EII products in the market are limited in, or totally lack, the capabilities of semantic interoperability and dynamic adaptation upon changes.

3 XML Technology – a Database Approach

XML documents can be either data-centric or document-centric. *Data-centric* XML is that which has record structure as its focus. Data-centric XML serves a similar function to a database; a set of fields are pre-defined, and records (think records here, not documents) must conform to that structure. *Document-centric* XML is that which has the document (the text, something pre-existing with its own structure) as its focus. A collection of XML documents can be conceived as an *XML database*.

Any access to XML data must be done through an XML data model. Traditional databases are based on the notions of a *database model* and a *database schema*. Elements, attributes, mixed content, and other features of XML do not give good assumptions for development of a unique model of XML data. For that reason different XML applications use different models of XML data, usually tree- or graph-oriented, or, more recently, combined with full text features. Perhaps the most important XML data model is that one used by languages XQuery, XSLT 2.0, and XPath 2.0. This model is richer than usual tree-like representation. In XPath 2.0, e.g., sequences replace node sets from XPath 1.0.

Solutions of many problems with manipulating XML data rely on a query language. We categorize XML queries into two classes: *databases queries* and *Information Retrieval (IR) queries*. Database queries return all query results that precisely match the queries, which reminds SQL querying in relational databases. IR queries allow “imprecise” or “approximate” query results, which are ranked based on their relevance to the queries. Only the top-ranked results are returned to users. Another proposal of W3C, rather delayed, is to effectively and efficiently update XML data (XQuery Update Facility).

One solution how to store XML data is to use conventional databases. It means to map (shred) the XML documents into data structures of the existing DBMSs (*XML-enabled database*). Detaching generic mappings of XML data into universal tables has the following properties:

- predefined schema is necessary,
- joins of tables are necessary for query evaluations and row ordering is done in an explicit way,

- navigations in XML data are transformed into SQL and full-text operations are also needed,
- scalability problems.

Another possibility is to store XML data into tables generated algorithmically from an XML schema.

A more advanced solution is to develop a DBMS with a native XML storage (*native XML database* or *NXD*), whose advantages include a support of:

- natural nested hierarchies,
- element ordering,
- documents as single objects,
- schema is not necessary,
- XPath and XQuery have a direct implementation,
- better scalability.

A *hybrid database* is a relational database that is XML-enabled, but also offers native XML capabilities as defined above. It is a database that supports both the relational data model and the XML data model in all its processing and storage mechanisms.

The XML technology relevant to XML databases concerns mainly XML schema and query languages. In the next two subsections we will discuss possibilities that both kinds of languages offer. Their choice in EII design can significantly influence the success of EII in practice.

3.1 Database Schemas and XML

By a schema we describe types of XML documents. In principle, two main possibilities are at disposal: DTD and XML Schema language. Current projects prefer schemas expressed in XML Schema.

XML Schema provides the ability to define an element's type (string, integer, etc.) and much finer constraints (a positive integer, a string starting with an uppercase letter, etc.). There is certain relationship between schemas expressed in these languages and database schemas. As in other DBMSs, an essential part of each schema definition languages is made by *integrity constraints*. Comparing to SQL in relational databases, possibilities of them in XML Schema are rather poor.

The specific problem is to design XML schemas. There are three ways to design XML schemas. The first, and the most difficult, is to attempt to create the schema directly element by element. This requires knowing in advance what specific elements already go where. The easier solution is to create an instance of the XML document, then use schema extraction tools to generate a schema that is valid for that instance. The last and most database-oriented possibility uses conceptual modelling XML data. Today, structure of XML data is designed usually directly, without the conceptual schema. This makes more difficult, e.g., modelling hierarchies like it is used in ER modelling. With XML conceptual modelling also automatic transformations to XML Schema are easier and more accurate. The research in this area is represented, e.g., by

[13]. Dynamicity of EII conditions requires yet an existence of tools for schema evolution and schema versioning.

Unfortunately, today's observation of EIS shows that requirements gathering, schema design and upgrade costs are far more than application development costs. A special feature of XML databases is that many XML documents exist whose schema was *not known* at design time. Thus, many vocabularies are developed without any schema. As a consequence there are XML databases without any schema. This fact belongs among the key reasons for existence of NXDs. Rather loose possibilities of XML schema development are much more flexible than relational or object-oriented structural definition. A sufficient compromise between completely schemaless and strict schema-oriented approach is to add cheaply and manageably a small amount of structure which provides a more compelling solution.

Often there is a need to extract the schema information from XML documents. The extracted schema should, on one side, tightly represent the data, and be concise and compact, on the other side. As the two requirements essentially contradict each other, finding an optimal trade-off is a difficult and challenging task. For promising results in this area see, e.g., [11].

3.2 XML Query Languages

XML query languages serve to querying, extraction, restructuring, integration, browsing XML data. They include the following demands:

- pattern matching,
- navigation along the structure of XML tags via (regular) path expressions,
- powerful approach to structured data similar to SQL,
- querying both data and metadata,
- generating structured answers to queries (new XML data, derived values)

There are a lot of standards in area of XML query languages designed by W3C, namely XPath 1.0 and 2.0, XQuery 1.0, XSLT 1.0 and XSLT 2.0. XPath 2.0 is a strict (rather large one) subset of XQuery 1.0. The main use of the XPath language is in other XML query languages, namely XQuery and XSLT. XSLT is a language of transformations. It provides instructions that help to transform XML data into a rendered format. The focus and strength of XQuery seems to be the data-centric queries (regularly structured markup), while XSLT has its advantages in document-centric queries (semi-structured markup).

Integration of relational and XML data resulted in development of SQL/XML language. SQL/XML allows relational data to be published in an XML form (XPath data model instance) that can then be queried using XQuery. It provides to define table columns of the XML type.

As the web-style searching becomes a ubiquitous tool, the need for integrating exact querying (see languages like XQuery, XSLT, SQL/XML) and IR techniques becomes more important. For example, in EIS environment we meet cases in which users provide keyword queries and require a ranking of partial results. In the case of XML, relevance scoring becomes more complex because the data required for scoring

have a tree structure. An attempt to integrate IR functionality with XQuery is described in W3C proposal [19]. For an excellent survey of XML retrieval see [14].

3.3 Architectures of XML databases: solutions

A significant role in storing XML documents is whether the documents are data-centric XML documents or document-centric XML documents.

As we have mentioned earlier, one possibility how to store XML data is an XML-enabled database. Experiments show that such database is most feasible if only simple XPath operations are used or if the applications are designed to work directly against the underlying relational schema. For similar reasons XSLT implementation can be based on use of a relational database, which serves as a temporal storage for source and target XML documents (e.g., [9]).

An implementation of NXD is undoubtedly a challenge in the last years both for developers and researches of database systems. In database architectures, NXDs provide a nice example when a DBMS needs a separate engine (see [15] for more deep discussion). There are three main approaches to NXD implementation today:

- NXD DBMS as a separate engine (Tamino, XHive/DB, XIndice, eXist, etc.),
- adding native XML storage to RDBMS (e.g., XML Data Synthesis by Oracle),
- hybrid solution (e.g., IBM DB2 9, ORACLE 11g, SQL Server 2008).

An advantage of the last two approaches is the possibility to mix XML with relational data. While critical data is still in a relational format, the data that not fit the relational data model is stored natively in XML.

With the new option of storing and querying XML in a RDBMS, schema designers face to the decision of what portion of their data to persist as XML and what portion as relational data. ReXSA described in [12] is a schema advisor tool that is being prototyped for IBM DB2 9, enabling to propose candidate database schemas given a conceptual model of the enterprise data.

Bourret in [4] registers more than 180 XML database products, among them 40 NXD, and more than 40 XML data binding products.

4 EII through XML technology

A motivation for maintaining XML data in databases has roots in application demands, in particular to ensure a better work with content in enterprises. With an XML database one can, e.g., process external data (Web pages, other text databases, structured data), resolve tasks of e-commerce (lists of products, personalized views of these lists, orders, invoices in e-commerce, e-brokering), and support integration of heterogeneous information sources. A typical example of the latter is an integrated processing data from Web pages and from tables of a relational database. There are XML database vendors who market their platforms as EII solutions (e.g., Software

AG, IBM, Ipedo). In other words, o store XML data in a database means to manage large numbers of XML documents in a more effective way.

Since storing and querying XML data as well as data integration are crucial for EII we focus on these kinds of NXDs uses in detail. We also mention Web services in context of EII and, finally, some EII trends.

4.1 Content Management Systems

It is well-known that reuse represents an important way for companies to extend the value of their investment in content. According to the study by ZapThink [22], producers of content spend over 60% of their time locating, formatting, and structuring content and just 40% for creating the content.

Stand-alone relational DBMSs are not well prepared for management such content due its unstructured nature. XML offers a robust technology that becane a background of *content management systems* (CMS). Such systems provide users tools for automatic conversion and distribution of native content via the Web. As XML separates formatting data from XML content, a new trend is to build CMS on the top of NXD. As a consequence the distinction between structured and unstructured information may now be blurring.

By SYS-CON Media Inc. [17], the following XML features are essential in the context of CMSs:

- *Content contribution and conversion.* Storing content in XML enables its various transformations into a variety of formats, such as HTML, for reuse by multiple applications.
- *Content access and exchange*. XML content can be easily merged with other sources and represented in an unified way in content management repository.
- *Content formatting and presentation*. A separation of content and presentation allows different formatting to be applied to the same content in different situation using XML stylesheets.
- *Content storage*. XML content stored in an XML database can be more easily searched by XQuery or XPath.
- *Content personalization*. Based on user profiles and type of device, CMS can deal with the content accommodated by an associate XSL stylesheet. Such tailored content is then delivered to the user.
- *Content management Web services*. Most of CMSs use Web services to share and deliver data and specific content management features in the Internet.

4.2 Data integration

XML databases have separated into three categories. The first one has focused on managing XML content or documents (e.g., MarkLogic). For example, MarkLogic Server provides a platform for CMS combining traditional DBMS based on XML with full-text searching. The other two categories are related to EISs. In the second

category, XML database can provide a middle tier *operational data store* (ODS) platform. In the third category, XML database focuses on managing persistent data on a middle tier for data integration applications, in particular EII applications (e.g., Ipedo).

Operational data store. A middle tier ODS can provide the necessary infrastructure for managing enterprise data and bringing it closer to the consuming business application, while simultaneously reducing the burden on backend systems of record. XML databases are an ideal technology to serve as an ODS because of their ability to maintain schemas and to bind heterogeneous data sources.

Enterprise information integration. Most current EII approaches are still based on similar principles of loosely-coupled federated systems. Moreover, a key issue, i.e. resolving differences in schemas and integrating them into one central schema, is often not requited in today's EII applications. XML databases enable EII by providing a platform for querying across heterogeneous data sources, resulting in view of all common entities spread across enterprise systems or services. For business users, typically, CMSs can become a source of integration in EII. In such systems data is managed as schema-less, eliminating the need for schema management and database administration.

4.3 EII and Web services integration

Web services create huge amounts of new data, specifically the exchange of data-rich XML messages. Many organizations want and need to store, access, query, audit, analyze, and repurpose information in these messages. It is nearly impossible to persist all of these messages in a relational database because of the inflexible data model they impose.

Here it is possible to use NXD as a „glue" to connect existing enterprise systems. For example, in SOAP XML-based object serialization format can be used to perform asynchronous messaging and RPC between non-XML applications. Although messages are probably data-centric, their natural format is XML. It makes sense to build a message queue on a NXD, particularly in cases when EII is *event-driven* rather than *query-driven*. Then data changes, for example, could be accumulated in a message queue and an EII query scheduled to run at periodic intervals to read the data from the queue and update a data store with the changes. We obtain XML-specific capabilities and, consequently, better scalability as the volume and complexity of e-business transactions increases. XML databases are particularly useful for handling new message types or evolving message structures. Storing message content in a native XML database reduces the development time and cost at least 50 percent by eliminating the need to define object-to-relational mapping [16].

4.4 EII trends

The ability to efficiently store and access XML and relational data types in one system represents a key point of differentiation among enterprise database vendors. It also allows enterprise developers to build data-driven applications using XML data

types. Open source RDBMS from companies such as PostgreSQL currently also supports this hybrid XML-relational data-storage capability.

The popularity of XSLT accelerated the EAI development, and some use of XSLT is probably a requirement in all EII solutions today. Any exchange of XML information is namely going to involve a combination of mapping of information objects, and in most cases these will involve structural transformations to account for different contexts, i.e., uses of the information.

Approaches to EII based on Semantic Web technologies like, e.g., RDF and OWL-DL ([2], [21]) are in development today. Authors of [2] use OWL-DL to integrate enterprise applications. Document content models are rendered into OWL-DL ontologies. This enables to designers to readily use automated reasoning methods of reasoners like e.g. RacerPro[4]. It means that a *model-driven* enterprise is achievable today.

5 Conclusions

EII is a broad area that in other terms and under other conditions restates problems studied in databases during the whole time of their existence. In [23] the authors address four challenges of EII including scalability, horizontal vs. vertical integration, central integration, and semantics. For example, with an increasing number of sources, the scale-up efficiency decreases. Integration is mostly horizontal than vertical in the most EII systems and the only vertical part is their centralized administration. We have seen that EII products fall into two categories, those that grew from an RDBMS background and those that emerged from the XML world. Particularly, users of hybrid RDBMS can profit from easier integration of structured and semistructured data. Really significant challenge is information sharing, which requires considering more semantics in EII. As we mentioned in the paper, techniques of the Semantic Web can contribute to this problem. According to prediction by META Group from 2005, ETL, EAI, and EII converge in the general Data Exchange Facility in the service-oriented architecture.

Acknowledgement This research has been partially supported by the grants of GACR No. GA201/09/0990 and P202/10/0761

References

1. Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, Y.-H., Simmen, D., Singh, A.: Damia – A Data Mashup Fabric for Intranet Applications. In: Prof. of VLDB ’07, Vienna, Austria, pp. 1370--1373 (2007)
2. Anicic, N., Ivezic, N., Jones, A.: An Architecture for Semantic Enterprise Application Integration Standards. In: Interoperability of Enterprise Software and Applications, Springer- London, p. 25--34 (2006)

[4] http://www.franz.com/agraph/racer/

3. BEA Systems: Liquid Data for WebLogic: Integrating Enterprise Data and Services. In: Proc. of SIGMOD 2004, Paris, France, pp. 917--918 (2004)
4. Bourret, R.: XML Database Products. http://www.rpbourret.com/xml/XMLDatabaseProds.htm (2009)
5. Fremantle, P., Weerawarana,S., and Khalaf, R.: Enterprise Services. Communications of the ACM October 2002/Vol. 45, No. 10, pp. 77--82, 2002.
6. Gilbane, F.: What is Enterprise Information Integration (EII)? The Gilbane Report: Volume 12, Number 6, Bluebill Advisors, Inc. © 1993 - 2005 The Gilbane Report (2004).
7. Hauch, R., Miller, A., Cardwell, R.: Information Intelligence: Metadata for Information Discovery, Access, and Integration. In: Proc. of SIGMOD Conf., Baltimore, Maryland, USA, pp. 793--798 (2005)
8. ISO/IEC 9075-14: Information technology -- Database languages -- SQL -- Part 14: XML-Related Specifications (SQL/XML) (2008)
9. Kmoch, O., Pokorny, J.: XSLT Implementation in a Relational Environment. In: Proc. of the IADIS Multi Conference on Computer Science and Information Systems - subconference Informatics 2008, Amsterdam, The Netherlands, pp. 91 -- 98 (2008)
10. Mattos, N.M.: Integrating Information for On Demand Computing. In: Proc. of VLDB'03, Berlin, Germany, pp. 8--14, 2003
11. Mlýnková, I., Nečaský, M: Towards Inference of More Realistic XSDs. In: Proc. of the 24th Annual ACM Symposium on Applied Computing - track Web Technologies, Honolulu, Hawaii, USA, ACM Press, pp. 632 -- 638 (2009)
12. Moro, M.M., Lim, L., Chang, Y-C.: Schema advisor for hybrid relational-XML DBMS. In: Proc. of the 2007 ACM SIGMOD Int. Conf. on Management of Data, Beijing, China, pp. 959-970, (2007)
13. Nečaský, M.: XSEM - A Conceptual Model for XML. In Proc. Fourth Asia-Pacific Conference on Conceptual Modelling (), Ballarat, Australia. CRPIT, 67. Roddick, J. F. and Annika, H., Eds., Australian Computer Society, pp. 37--48 (2007)
14. Pal, S., Mitra, M.: XML Retrieval: A Survey, Technical Report, CVPR, 2007, TR/ISI/CVPR/IR07-01, (2007)
15. Pokorný, J.: Database Architectures: Current Trends and Their Relationships to Requirements of Practice. In: INFORMATION SYSTEMS DEVELOPMENT Series, Advances in Information Systems Development: New Methods and Practice for the Networked Society, Springer Verlag, pp. 269--279 (2007)
16. Smik, R., Parikh A., Ramachandran, A.Use XML databases to empower Java Web services - Integrate a native XML operational data store into your enterprise application JavaWorld.com, pp- 1--7 (2004)
17. SYS-CON Media Inc.: The Role in XML in Content Management. XML Journal, (2008)
18. Tailor, J.T.: Enterprise Information Integration: A New Definition. Integration Consortium, DM Review Online, September 2 (2004)
19. W3C: XQuery and XPath Full Text 1.0 Requirements. W3C Working Draft, (2008)
20. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M. : A Framework for Rapid Integration of Presentation Components. In: WWW 2007, Banff, Alberta, Canada, pp. 923--982 (2007)
21. Yuan, J., Bahrami, A., Wang, Ch., Murray, M., Hunt, A.: A Semantic Information Integration Tool Suite. In: Proc. of VLDB'06, Seoul, Korea, pp. 1171--1174 (2006)
22. ZapThink: Market for XML-enabled Content Lifecycle Solutions to Exceed $11.6 Billion by 2008; XML Key to Solving Critical Content Management Problem: Content Reuse. Business Wire (2003)
23. Zhou, J. Wang, M., Zhao, H.: Enterprise Information Integration: State of the Art and Technical Challenges. Proc. of PROLAMAT, IFIP TC5 International Conference, pp. 847--852 (2006)