

Standing on the Shoulders of Giants: Native XPath enabled storage and retrieval with MySQL

Thomas Nicolai, Marc-Frederic Schäfer, Lars Kirchhoff

Institute for Media and Communications Management
University of St.Gallen
Blumenbergplatz 8
9000 St.Gallen
thomas.nicolai@unisg.ch
marc-frederic.schaefer@unisg.ch
lars.kirchhoff@unisg.ch

Abstract: XML has become a standard for the exchange and retrieval of data between different platforms but is not currently a standard in storing XML documents in relational databases. A standard implies wide acceptance and usage. In this paper we present a new Open Source approach towards native XML support in relational databases especially for the common and widely used MySQL database system¹. We have developed a digital multilingual research platform which uses native XPath database support. We show means of input, storage and manipulation of XML documents in relational databases and contribute new insights to the document retrieval research area. Therefore we present the design of the platform and finally give an outlook towards further opportunities implicated by our approach to the information retrieval scientific community.

1 Introduction

For platform independent representation and retrieval of information and especially for the exchange of information the eXtensible Markup Language (XML) [1] has become a standard [2]. The information exchange between platforms is often implemented via web services provided by an application interface. For this purpose XML documents are created by the underlying system from primary data stored in relational databases. XML documents are just a transient representation of the information exchanged between different types of applications [3].

¹ From an email on 24.03.2005 between Thomas Nicolai and Alexander Barkov {bar at mysql dot com}; „I'm looking forward to having feedback from you, you're the first user of this code :) “. We would like to take this opportunity to again formally thank Alexander Barkov and MySQLAB for their friendly and guiding support during all this hard time in the last year.

On the other hand Markup Languages like SGML [4] and XML have been widely used for representing structured documents (complex units of information, consisting of formal and natural languages) [3]. By using XML for the representation of information a persistent storage of XML documents is mostly not necessary as the XML format can be translated into a relational data structure [5]. Therefore the XML document has to fit the relational structure of the database design. Changes in the XML document structure have direct implications for the underlying relational database structure. One of the main problems for data exchange between various applications is the consistent and uniform management of data via standardized XML structures.

Especially the increasing disintegration of media and content influences the design of platform architectures, especially when these have been designed for the processing and relaying of digital content [6, 7, 8]. The more information saved digitally, the wider the array of data formats, data structures and data types, as well as their newly arising (usage-) possibilities. For the exchange of this variety of different data structures, especially their meta-data description protocols are defined to avoid media breaks and expensive human interaction. For this reason in recent years standards like Dublin Core [9] or the OAI-PMH [10] protocol for the exchange of meta-data between digital libraries have been created. Most of these standards are based on XML documents by which information exchange is quite simple. But by translating the XML documents into a relational structure to save them in a relational database much of the XML's inherent flexibility is lost. Especially if more information than described by the used standards is provided it may not fit with the underlying relational database model and thus cannot be stored within the relational database without adapting it. A mean to avoid this problem is to find effective means to store and manage persistent XML data in databases.

One of the main difficulties in using traditional database technologies for the storage and management of complete XML documents is based on its specific characteristics [3], especially in their flexibility. The accommodation of XML document structures is quite easy. But exactly this causes major problems for a relational database model, as the model and thus the database structure as well as the application structure needs to be changed.

2 Motivation

Such difficulties could be avoided if XML documents are persistently stored in a relational database as a whole. On the other hand the handling of persistent XML documents in relational databases entails problems to store, access, and manipulate XML documents within databases. Different approaches exist to process XML documents in combination with relational databases [11, 12, 13]. These approaches are mainly distinguished by two different types of XML documents. These are document-centric and data-centric XML documents [14]. Due to their mostly irregular and variable structured document-centric XML documents require that their structure is preserved as a whole and therefore this kind of XML documents are normally stored as a whole in one table column. Data-centric XML documents mostly contain structured information which can easily be decomposed and stored in relational databases. Therefore data-centric XML documents are mostly "shreddered", so that each data element is stored in a specific column of a specific table within this database [5]. For retrieval and exchange purposes the XML document is then recreated which has a large impact on the time and resource consumption.

3 Theory

To exchange data between various platforms mainly two possibilities exist. Either the exchange of data is performed via a file (e.g. XML) which is exported from system A and imported into system B. For this process mainly a transformation application is needed. The second possibility is to integrate data from various sources via a global data schema by using a mediator.

3.1. Tradition database design

As the database structure normally cannot be changed easily, the data structure of application has to be stable. A change in the data structure has to be followed by a change in the database structure, because the relational database model has to be an exact mapping of the data structure. Of course it is possible to create new tables and columns by SQL statements, but the automatic adjustments of the database architecture are delicate. Therefore the database architecture is normally modified and cleaned out manually. To exchange information via different platforms mainly three types of heterogeneities have to be overcome [19]:

- heterogeneities of the database applications
- heterogeneities of the syntax and relational structure
- heterogeneities on a semantic level

The main difficulty is to generate automatic links between the different definitions on each to the three types of heterogeneities. For the data exchange between different platforms the first type normally does not cause any problems. For web applications it depends on the database query language (e.g. SQL) to access the data stored in the database. For the third type of heterogeneities it is possible to create or use ontology which defines equivalence and relations between different data models on a semantic level. To overcome the heterogeneities of the syntax and relational structure often mediators are used to create a common global data schema. To overcome this problem we propose to abstract the data model from the database model.

3.2. Abstraction from the data structure

One possibility to avoid inconsistencies on the database architecture is to separate the database structure from the data structure (syntax). This can be achieved by representing the data structure in XML documents and afterwards storing them within the relational database as a whole. The relational database model focuses on relations only needed for the application layer independent of the data structure of contents used on the platform. For example the storage of publication metadata within a relational database would consist of the storage of the XML document (content object) in a table of the database with a unique identifier. For managing the access rights which are used by the application layer, a relation between the content object and the table where the access rights and users are stored is established. The information for such a relation is extracted from the XML document itself. By extracting the data which is relevant for the relational structure of the database (which is only determined by the structure of the application itself) from the XML document and storing all other information as an XML document in the database a perfect abstraction from the data structure (syntax) can be achieved. This enables database designers to optimize the database structure along the needs of the application layer independent of the data structure processed on the platform. A key requirement to be able to perform such an abstraction on the database layer depends on the possibilities to process XML documents within the underlying database application.

3.3. Means of storing and retrieving XML data from databases

The problem of storing XML documents in a relational database is well known and various approaches exist to solve this problem [11, 12, 13]. In the following three generic types of storing XML documents in relational databases are presented.

3.3.1. XML "shredding"

This technique mostly focuses on decomposing the XML document and storing the data contained in tables, rows and columns of a relational database [3]. For separating the data structure from the database structure this technique is not adequate. But by using this mean XML documents can be stored within a relational database and they can be queried without performance loss as the full potential of the relational database can be used. On the other hand the decomposition of XML document requires a stable data structure. Otherwise the decomposition algorithm and the database structure have to be adjusted at each change of the data structure.

3.3.2. XML parser in combination with a relational database

With this technique XML documents are stored in a relational database as a whole. For querying the documents are retrieved from the relational database and processed via a XML parser (DOM, SAX) [11, 12]. XML documents can be stored within the relational database as a whole and therefore an abstraction of the database model from the data model is possible. A change in the data structure has no effect on the database model, as the data structure is only included in the XML document. The main disadvantage of this solution is that parsing XML documents is very time consuming and cost intensive. Especially if the number of documents or the complexity of queries grows the performance of parser degrades heavily.

3.3.3. Native XPath / XQuery processor of the database application.

XPath is a query language for XML documents by which specific nodes within a XML document can be addressed directly [16]. The integration of a native XPath processor in the database application enables the querying of XML documents within the relational database. With this technology both normal data and XML documents stored within relational databases can be queried by SQL statements with the performance of a relational database. This technology makes it possible to separate the database structure from the data structure. But the technology is very new and not many standard database applications provide a native XPath support. In its latest version (Version 5.0.5dev with XPath patch) MySQL for the first time provides a native XPath processor [23]. The fully integrated version shall be released with version 5.1.

4. Application of the separation of data and database structure

The following case study shows an implementation where the data model is separated from the database model to optimize the interoperability (e.g. web services, different media channels), adaptability and reuse as well as to enhance the retrieval functionalities of the research platform Alexandria.

4.1. The Alexandria research platform

The Internet has become medium for everyday work and information research and it is now hardly possible to imagine daily life without it. Although search engines like Google or Yahoo offer far reaching search possibilities it is decidedly difficult and time consuming to find qualitatively valuable sources of information. Most notably for scientific research, the problem of being able to trust the scientific accuracy of the search results often arises. Hence, the search turns out to be very complicated not just when looking for concrete documents but content.

With the burgeoning volume of internet sources and data, it becomes therefore even harder to differentiate between trustworthy, legitimate information and other kinds of information. In order to protect against overload of information, an aggregation of specific types or kinds of information, for example scientific research, is necessary. This same issue also holds true for the professional face of research at the University of St. Gallen. Each institute possesses its own homepage, through which scientific results are published. If two institutes are working in similar areas of research, there is unlikely to be a link between the two web pages, due to the difficulty of devising research-content theme-based searches. So that the University of St. Gallen would have at its disposal its own standardized access to the scientific activities and research results, the University of St. Gallen developed the Alexandria research platform. Under the leadership of the rectorates, the Alexandria Project was initiated in January 2004. With the official start of the research platform in April 2005, there is now standardized access to all research results of all of the institutes of the University of St. Gallen.

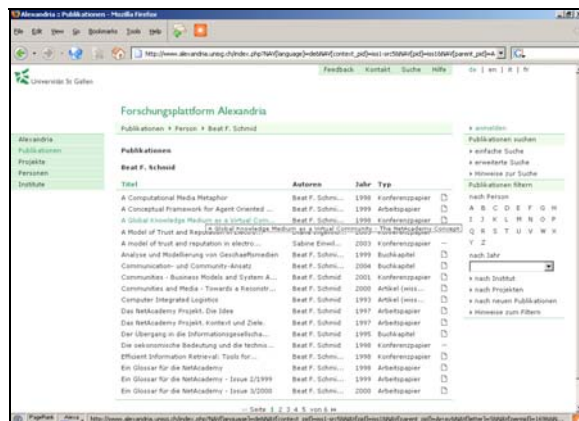


Figure 2. Screenshot of the Alexandria research platform [19]

The platform is based on the Netacademy platform developed in 1997 at the Institute for Media and Communication Management that was further expanded to suit the needs of the University of St. Gallen.

4.2. The Approach

For the development of the Alexandria research platform the kinds of XML content are of central importance, since they make up the heart of the platform, and constitute the starting point of the entire implementation. The conception is based on an object oriented consideration of these content types, which is made possible by a later expansion.

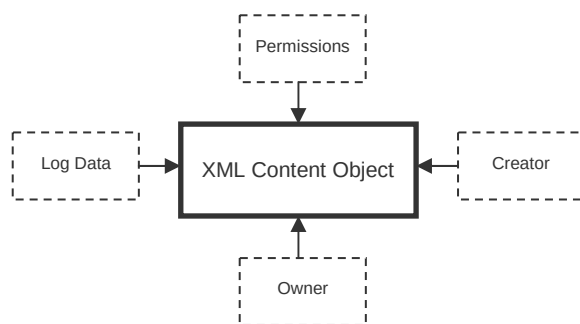


Figure 3. Object-oriented observations of the XML content objects

Each XML content object is coupled with its platform-based characteristics (from the logbook data entry to the user management and the authors (Creator after Dublin Core [9])). Figure 3 illustrates the interconnections within the database, which are mapped by means of a relational database model². For the classification of XML content objects in the database design it is important to easily identify the overlapping tables in an index. In this context the relationships between various XML content objects are of particular relevance, since these are partly not mapped in the relational database and thus in this regard the benefits of a relational database are not used. Instead they are stored in the database partially added within the XML data. In future this disadvantage could be overcome in using the XPath 2.0 standard element for external references [16].

² More exacting observations may be gathered from the presentation of the Swiss PHP Usergroup, since here a focus on the XPath observations should be taken up.

4.2.1. Rough concept: From the form generation to saving XML data.

Based on the experience of Digideep.com [20] the text document based approach was expanded to the XML format. This was done by rendering the XHTML forms from XML documents, where a field in the document is generated from each node of the first level, and where the characteristics are defined from the next generation.

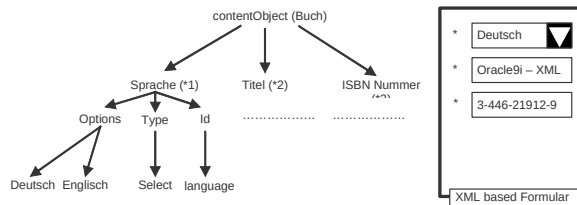


Figure 4. XHTML generated Form on a basis of the content objects which are described through XML documents

The documents serve thus not only to define the XHTML forms but they are simultaneously the specifications of the to-be-saved metadata. In this case, the data that is returned via a parser after being entered is compared to the content object defined XML document. This procedure has two advantages:

- The automatic removal of foreseen not-to-be-saved elements like “hidden fields” that exclusively serve the “processing” of the forms.
- The checking of how the fields are allocated as well as their worth based on their specification and type.

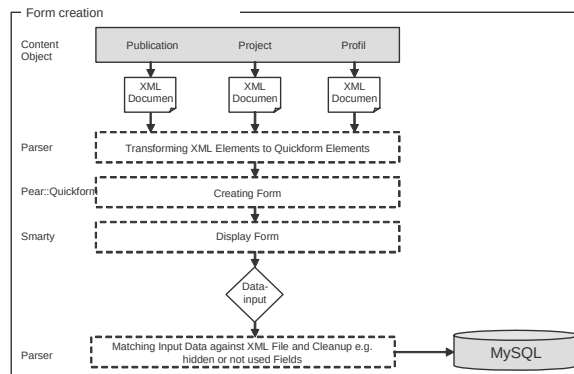


Figure 5. XML form processing³ [17]

³ Smarty is a PHP based template engine that separates content from presentation [18]

Figure 5 shows the individual steps from the rendering of the XML document to the saving of the XML data in the database. In the first step the content object defined XML document is transformed into PHP Pear::Quickform compliant objects [21]. These are then displayed using the Smarty Template Engine and can be filled out by the user [22]. Finally the returned delivery formula data are compared against the XML source document and all of the unnecessary data is removed. The in turn newly (cleaned up) resulting XML document is then filed in the MySQL database.

4.2.2 The Concept in Detail: XPath based solution. The prerequisite for the database design is the expandability pertaining to new content types as well as pertaining to the metadata of a content object. The database design and table design is shown in figure 6. Here the conversion of every content object type is always based on four simultaneously present fields. For this reason it is possible to map without difficulty all new content object types.

contentObject_id	createTime	updateTime	language	xml_data
78	2004-10-09 19:10:01	2005-04-20 08:47:27	fr	<?xml version="1.0" encoding="ISO-8859-1"?> <contentObject> <type>journal_paper</type> <title>La gestion des innovations et des technologies: un facteur de succès</title> <journal_title>La Vie économique</journal_title> <pages>26-28</pages>

Figure 6. Table structure for the saving of content objects

The “xml_data” field is presently a normal text field. An XML data type doesn’t yet exist for MySQL. It is still in planning and should be integrated into Version 5.1 by release time. The syntax for the node access of XML documents in the MySQL database takes place using the Xpath standard [16], where presently in the context of the unofficial patch almost 90% of the standards are implemented [23].

The elimination, for example of the title of the publication shown in figure 6, is achieved with the following XPath command:

```
mysql> select ExtractValue(co_object_data, '/contentObject/title') from cO_publications
where contentObject_id = '78';
```

```
+-----+
| ExtractValue(co_object_data, '/contentObject/title') |
+-----+
| La gestion des innovations et des technologies: un facteur de succès |
+-----+
```

The use of the *extractValue(xml,xpath)* command is also in the SQL-Statements section possible for every user-defined variation:

```
mysql> select ExtractValue( co_object_data, '/contentObject/type') from cO_publications
where ExtractValue(co_object_data, '/contentObject/type') = 'journal_paper';
```

```
+-----+
| ExtractValue(co_object_data, '/contentObject/type') |
+-----+
| journal_article |
+-----+
```

Next to the `extractValue` command XPath support also offers an *`updateXML(xml,xpath,content)`* command in order to change individual data within an XML document:

```
mysql> update cO_publications SET co_object_data = updateXML(
co_object_data,'/contentObject/title/.','<title>New title after XPath update query</title>'
) where contentObject_id = '78';
```

```
mysql> select ExtractValue(co_object_data, '/contentObject/title')
from cO_publications where contentObject_id = '78';
```

```
+-----+
|ExtractValue(co_object_data, '/contentObject/title') |
+-----+
|New Title after XPath update query                  |
+-----+
```

4.3. Critical consideration

Basically after a basic performance analysis the subjective impression of the performance cannot be confirmed⁴ for larger amounts of data records. Presently more than 15000 XML data records are located in the database, the performance of which has been consistently positively valued by the user in contrast to the large scale measurements. Since as a rule only individual data records are updated, this is then the insight advantage, which according to the user survey is harder to value. Furthermore a number of observations were made regarding the implementation of the research platform. The most important of these are described as follows:

1. Often because of the design approach, the saving of relations in XML documents can present structural problems and unwanted complexities.
2. The architecture has shown that through the selected structure and database design, independence from the saved data can be configured thus making the interoperability (layered architecture) with other systems much easier.
3. Changes to the data structure of the individual content objects require no alteration to the database structure. This abstraction has also had positive influence on the counter-usability of the classes and functions of the PHP source codes.
4. The premature assessment of an established XML schema (like for example Dublin Core) makes a later adaptation of this redundant. These eliminate adaptation or a later adjustment of the XML document and the XML data in the database.

⁴ From a subjective impression no tangible performance difference can be ascertained by working with the Alexandria research platform. Further observations are a subject matter for future research.

5. Since there are currently no management tools (like PHPMyAdmin) for the support of XPath, the direct manipulation of the data is highly difficult.

Finally, inspite of the still considerable barriers, very positive results can be drawn from the experience of implementing the platform. The performance and flexibility, which is reached through the native XPath support, is for the goals of the platform absolutely sufficient. Furthermore the as-a-result-of-the-design targeted variability can be completely changed. The image of relations between content objects is still now not optimally solved, and surely presents a further development potential for the future. In addition, changes to the XML data structure to an already established standard, like for example, Dublin Core Standard appear usually important, in order to avoid mapping between or transformation to different XML standards in the future.

5. Towards Information and Document Retrieval Scientific Discussions

As the research platform already supports the Dublin Cores standard in XML [26] it has several advantages to include systems that use search functionalities like the XRank Keyword Search [24]. Basically from an information retrieval viewpoint we can show by four reasons that our approach will have a major impact towards the actual discussions:

1. Provision of additional semantics: As XML documents are stored in the database extensible element tags are allowed. Through this it is possible to capture additional semantics which can not be captured with fulltext searches through databases or HTML documents (on the presentation layer) [27].
2. Granularity of XML Element: In contrast to HTML documents XML documents have a hierarchical structure. Due to this XML keyword search queries can address single XML elements containing the desired keyword. As our approach is based on a very popular web platform (PHP, Apache & MySQL) new retrieval concepts like XRank may become a standard besides normal HTML fulltext search.
3. Nested XML: Rankings can be based on the hierarchical structure and thus on the granularity of XML Documents. In contrast to typical search Engines like Google which are based on the hyperlink structure [24, 27] XML keyword search engines can return nested elements and thus enhance the quality of search results.
4. New retrieval Interfaces: Due to the hierarchical structure of XML documents different models and interfaces than normal fulltext search approaches can be implemented. The main advantage would be that it would not only be possible to query, browse and refine query based on full documents only but also on specific parts or elements of them [25].

By using our approach, which not only provides a huge complexity reduction in programming web applications but also provides an until now not known flexibility in the data structure in working with widely used relational databases, the provision of structured documents for information retrieval lies at the hand. The combination of XPath and SQL allows the development of better HCI and allows enhancing the performance of query processing as well.

6. Conclusions and Future Work

The possibilities to use XPath in combination with SQL and as a consequence thereof the possible separation of the data structure from the database structure offer new and powerful opportunities for information retrieval. In our approach we show how heterogeneities of the syntax and relational structure can easily be overcome by storing complete XML documents in a relational database. This has among others three main impacts:

1. It is possible to retrieve and manipulate native XML documents in the relational database directly without performance losses through XPath. This results in more accurate search results compared to full text searches as single nodes of the XML document can be addressed directly.
2. The abstraction between the data model and the database model provides complexity reductions for database designers. By using this technique, the database architecture only needs to match the requirements of the above lying application and is independent of the data structure.
3. It is possible to interchange XML documents between various applications without the need of import and export filters. The combination of XPath and SQL provide very powerful means for querying and retrieving stored data systematically.

The results presented in this paper are research in progress. The intention is to introduce the novel concept to store and retrieve complete XML documents using a native XPath support for relational databases. Within this context further developments will be undertaken, e.g. the elimination of relations in XML documents to avoid structural problems. Further developments like the integration of access restriction possibilities which are embedded in the XML documents (by using XACML [21]) or the introduction of ontology in combination with used XML schemas are evaluated and perhaps implemented in the future to facilitate cross media data exchange and retrieval. Within this paper we have shown that it is possible to theoretically overcome barriers of different database schemas for cross media information retrieval by using native XPath functionalities.

7. Acknowledgement

We would espacially thank Alexander Barkov and MySQLAB for their kind support during the development of the research platform Alexandria. Furthermore we would like to say thank you to the PHP and Pear Community for their developments.

References

- [1] W3C (2005): eXtensible Markup Language (XML), <http://www.w3.org/XML>
- [2] Edwards, R., Hope, S.: Persistent DOM: An architecture for XML repositories in relational databases, http://www.sees.bangor.ac.uk/~rich/research/papers/uwb_rge_IDEAL2000.pdf
- [3] Salminen, A., Tompa, F.Wm.: Requirements for XML Document Database Systems, <http://db.uwaterloo.ca/~fwtompa/.papers/xmlldb-desiderata.pdf>.
- [4] W3C: On SGML and HTML, <http://www.w3.org/TR/REC-html40/intro/sgmltut.html>
- [5] Shanmugasundaram, J., Krishnamurthy, R., Tatarinov, I.: A general technique for querying XML Documents using a relational database system, <http://www.cs.cornell.edu/people/jai/papers/GeneralXMLQuerying.pdf>.
- [6] Zerdik, A., Picot, A. Scharpe, K.: E-Merging Media, Springer Verlag, Berlin, 2004.
- [7] Schumann, M., Hess, T.: Grundlagen der Medienwirtschaft, Springer Verlag, Berlin, 2000.
- [8] Schumann, M., Hess, T.: Medienunternehmen im digitalen Zeitalter: neue Technologien - neue Märkte - neue Geschäftsansätze, Gabler Verlag, Wiesbaden, 1999.
- [9] Dublin Core Metadata Initiative: Dublin Core Metadata Element Set, Version 1.1: Reference Description, <http://dublincore.org/documents/dces>
- [10] Lagoze, C., Van de Sompel, H.: Open Archives Initiative - Protocol for Metadata Harvesting - V 2.0, <http://www.openarchives.org/OAI/openarchivesprotocol.html>.
- [11] Edwards, R., Hope, S.: Persistent DOM: An architecture for XML repositories in relational databases, http://www.sees.bangor.ac.uk/~rich/research/papers/uwb_rge_IDEAL2000.pdf
- [12] Nicola, M., John, J.: XML Parsing: A Threat to Database Performance, http://lists.w3.org/Archives/Public/www-ws/2004Oct/att-0032/MNicola_CIKM_2003_1_.pdf
- [13] Klettke, M., Meyer, H.: Managing XML documents in object-relational databases, <http://citeseer.ist.psu.edu/cache/papers/cs/14668/http:zSzzSzwww.informatik.uni-rostock.dezSz~meikezSzxml2000.pdf/klettke99managing.pdf>
- [14] Mlynkova, I.: Overview and Possible Improvements of Techniques for Processing XML documents in (O)RDBMS, <http://kocour.ms.mff.cuni.cz/~mlynkova/doc/XMLday2005.pdf>
- [15] Wikipedia: OSI Model: http://en.wikipedia.org/wiki/OSI_model

- [16] W3C: XML Path Language (XPath) 2.0: <http://www.w3.org/TR/2005/WD-xpath20-20050404/>
- [17] Nicolai, T.: Forschungsplattform Alexandria: Technologisches Designkonzept; Präsentation PHP Usergroup Schweiz: <http://www.alexandria.unisg.ch/publications/18475>
- [18] Smarty Template Engine (2005): <http://smarty.php.net>
- [19] Hansen, H.R., Neumann G.: Wirtschaftsinformatik 2 Informationstechnik, Lucius & Lucius, Stuttgart, 2005
- [20] Alexandria Research Platform: <http://www.alexandria.unisg.ch>
- [21] Pear Quickform Library: <http://pear.php.net>
- [22] Smarty Template Engine: <http://smarty.php.net/>
- [23] Alexander Barkov: XPath Presentation, <http://mysql.r18.ru/~bar/myxml/XMLXpathSupportInMySQL.pdf>
- [24] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram: XRANK: Ranked keyword search over XML documents. In ACM SIGMOD, 2003.
- [25] F. Crestani, J. Vegas and P. de la Fuente: A Graphical User Interface for the Retrieval of Hierarchically Structured Documents. Information Processing & Management, 40(2):269-289, 2004.
- [26] A. Powell: Guidelines for implementing Dublin Core in XML. 2003. <http://dublincore.org/documents/dc-xml-guidelines/>
- [27] Guo, L., Shao F., Botev C., Shanmugasundaram, J.: XRank: Ranked Keyword Search over XML Documents, <http://www.cs.cornell.edu/People/jai/papers/XRank.pdf>