Exploring Statistical Information for Applications-Specific Design and Evaluation of Hybrid XML storage.

Lena Strömbäck, Valentina Ivanova, David Hall
Department of Computer and Information Science
Linköpings Universitet
S-581 83 Linköping, Sweden
lena.stromback@liu.se, valentina.ivanova@liu.se, david.hall@liu.se

***Abstract* — Modern relational database management systems provide hybrid XML storage, combining relational and native technologies. Hybrid storage offers many design alternatives for XML data and in this paper we explore how to aid the user in effective design of hybrid storage. In particular we investigate how the XML schema and statistical information about the data can support the storage design process. We present an extended version of our tool HShreX that uses statistical information about a data to enable fast evaluation of alternative hybrid design solutions. In addition we show the benefit of the approach by a first evaluation where we discuss how the tool aids in the storage design and evaluation process.**

Keywords – XML, Hybrid XML management, indexing, storage design

I. INTRODUCTION

The rapid increase in web based applications yields an increasing interest in using XML for representation of data. XML is able to represent all kinds of data ranging from marked-up text, through so called semi-structured data to traditional, well structured datasets. Supporting the flexibility that makes XML appealing is challenging from data management and technical perspectives. Several approaches have been used including native databases and shredding XML documents into relations. In practice, hybrid storage that combines native and relational solutions is of large interest. Hybrid storage is provided by the major relational database vendors (Oracle, IBM DB2 and Microsoft SQL Server). They offer interesting options for storage design where native and relational storage can be used side by side.

Several studies evaluate different solutions for XML management (e.g., [22][24][26][31]). For shredding, it is well known that the choice of translation strategy affects the efficiency (e.g., [5][8][10][12]) while hybrid XML storage, has so far only been studied in a few cases, (e.g., [16][17][27][28]). The above studies discuss a number of features that may have an impact on how to achieve efficient storage; the complexity and regularity of the XML structure; how the data is queried, i.e., the access patterns for different entities in the data set; and the frequency of references to other sources.

In this paper, we further explore these issues by investigating the impact of the application on the performance of the database. The properties we are focusing on are the XML schema structure and statistical properties of the data set. We first give some further motivation and discuss the goals of our work. This is followed by a discussion of properties and measurements relevant for storage design. We then present a tool that enables fast evaluation and exploration of storage solutions and present a first evaluation to show the feasibility of the tool. The paper is summarized by presenting our future vision. Our long term goal with the work is to present a method that can suggest a set of plausible hybrid storage models for an application.

II. MOTIVATION AND GOALS

Previous work (e.g., [1][3][5][8][23]) have defined efficient shredding methods for XML data into relational databases that result in fast query times. For hybrid storage the situation is more complex where an inappropriate choice of storage design can lead to poor performance [25]. In general automatically shredded relational XML mappings can lead to a rather large and complicated structure of relations. On the other hand, storing entire XML documents natively in XML storage keeps the structure completely intact to the cost of slow access to the data. For hybrid XML storage we have the choice to store parts of the XML structure as relations and other parts as XML and can gain from the benefit of a good data model and relatively fast performance. The design of a good hybrid storage model is complex and dependent on the requirements for the specific application [25].

Exploration and evaluation of alternative solutions is a time consuming task. Methods and tools, to aid the user in design of hybrid storage, and measurements, that could give hints on how to make choices, are of high importance. In a preliminary evaluation we compared the query efficiency with the amount of data stored as XML in the hybrid solution. In our tests, we adopt the shredding principles used in ShreX [1][6] as these principles give a mapping that captures the semantics of a given XML schema for the XML data. To explore hybrid storage we used the extended system HShreX [27][29], which also allows hybrid XML mappings. The general principle behind the mappings of these systems is that complex elements are translated to relational tables. Simple elements and attributes are shredded to a column in their parent table if they occur at maximum once in its parent element. HShreX extends this basic shredding by providing

 ISBN:978-1-61208-115-1

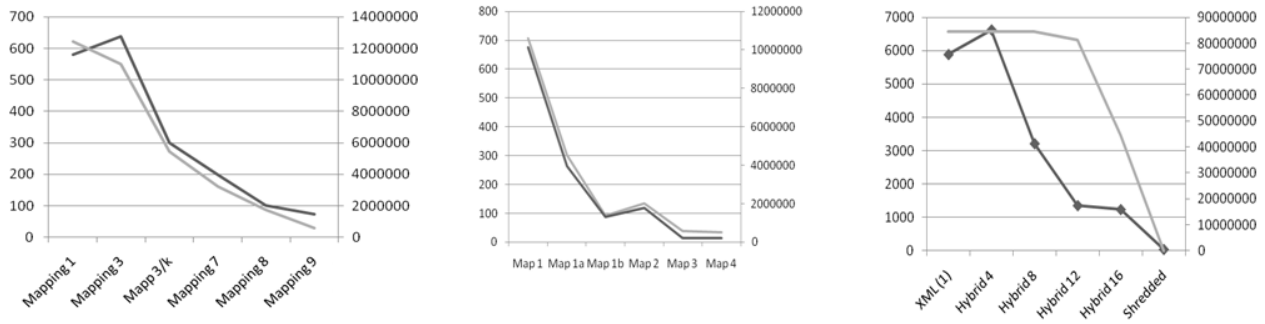

Figure 1. Run times [ms] (black) and data size [bytes] (grey) for PSI-MI (left), UniProt (middle) and Michigan Benchmark (right)

hybrid XML storage, i.e., to allow parts of the structure to be kept as XML in the final database representation. In our study the complexity of the created models varies between one or two relations for the models stored in pure XML to over 100 relations for the fully shredded data models.

The results of these tests are illustrated in Figure 1. The first two graphs show the results for two real data sets from the IntAct [2] and UniProt [30] databases. In this case we can see that the amount of data stored as XML gives a good estimation of the expected query time. For the Michigan Benchmark data [19] the estimation is not as good as for the two other datasets. This means that the amount of data is a good indicator for the performance, but also that further statistics about the data could give us better indicators and aid in effective storage design.

III. AVAILABLE INFORMATION

In principle there are three sources of information that can be used to learn more about the features and storage requirements of a computer application. These are; The general data schema, i.e., the data model; Samples of data to determine how the data model is used and what parts of the data model are in most common use; Samples of queries to determine what kind of queries are often performed for the data. In this work, we will examine how to use the data model and statistical information for a particular dataset.

As shown in the previous section the amount of data stored as XML is related to the query performance. However, the prediction we get from simply measuring the amount of data is not enough, we also need to collect more detailed information about the structure of the data. In practice, different parts of the XML schema are populated differently in different data sets. The XML schema carries information about the general structure, but, as for relational databases, the schema does not give a full picture of how this structure is instantiated for a particular dataset. We want to capture this information to create an effective hybrid storage model. In previous work [13], where we worked with generated data, we could see that also the amount of data at various positions in the XML file and the structure of this data had an impact on query performance. We wanted to explore this further and collected the following information:

- Overall statistics for the dataset. With this we mean characterizing the general structure of the dataset. For this purpose we use simple measures, such as, the total number of attributes, elements, and levels in the XML. We also collect the number of elements at each level of the dataset to determine the fan out of the data.
- Diversity of the dataset. To get estimations of diversity we collect the number of elements and attributes for each element or attribute string, at which depths they occur and compare those to the number of overall elements. We also collect information on how many unique search paths occur within the data set and the number of their occurrence.
- Detailed information at each position in the file. This is collected by counting the occurrence of element names at each level in the file. For each combination of parent/child node we count how common the child node is for this parent and collect the minimum, maximum and mean number of times this child occurs for the parent.

Our work on generated data has shown that parent/child statistics were of particular interest since this had a large impact on query performance. Figure 2 shows an example of the parent/child statistics. In the XML schema tree we show how common the different child nodes are in the parents.

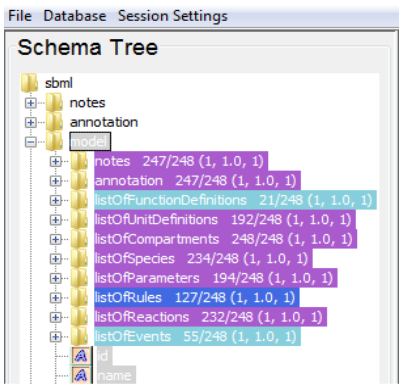

Figure 2. Statistics in HShreX

IV. A TOOL FOR EVALUATION

To allow easy access to the statistics and aid in evaluating storage alternatives we extended our tool HShreX to include this new information. The new version of the tool can be used to create and evaluate different XML storage models. The system analyses an XML schema and represents it as a tree structure, which facilitates its visual perception. The tree structure helps to easily understand and navigate the schema components as well. The relational schema, which the HShreX user can create in a database, is likewise created during the schema analyses. Once the database structures are created, large datasets, which corresponds to the currently parsed schema, can be quickly shredded in the database. Each step starting from the XML schema parsing and ending in datasets loading is logged and available for review in a panel under the main work area.

The relational schema is created following the shredding strategy, mentioned above. The user can alter the data shredding rules using HShreX annotations [27]. In this way, the XML data can be represented in purely native, mixed and shredded storage models. The HShreX annotations provide the opportunity to switch rapidly and flexibly between different storage models, create them in a database and evaluate their performance features.

HShreX's user interface provides three panels, which give more details of the schema elements and their mappings. The first panel lists specific details, such as currently applied HShreX annotations, children elements and attributes and their occurrences, for the currently selected element in the XML schema tree. The second shows HShreX mapping of the selected element or attribute in the tree. The relational tables and their relations are available in figures in the third panel.

In this work, the user interface was extended in two directions – to provide more convenient work with HShreX annotations and to visualize more information for a particular dataset. Figure 3 shows the dialog that facilitates manipulation of HShreX annotations. While navigating in the schema tree, we can open the dialog for the element or the attribute of interest and process its annotations. The dialog provides functionality for adding annotations, updating, i.e., changing values of available annotations and deleting annotations. Since some combinations of annotations for an element or an attribute are not valid, we validate each annotation regarding the already available annotations prior to adding. A useful feature is provided through the "Apply all changes to all elements of this type" button i.e., the currently added/removed annotations will be applied to all elements of this type in the XML schema with a single action. The basic data and the annotations, which apply to the element or the attribute of interest, are listed in the right side of the dialog.

The second improvement in the user interface is orientated towards the statistical information available for a particular dataset. HShreX obtains this information by analyzing a set of sample XML files representing the dataset. Detailed information, for the element or the attribute of interest and its children elements and attributes, is presented

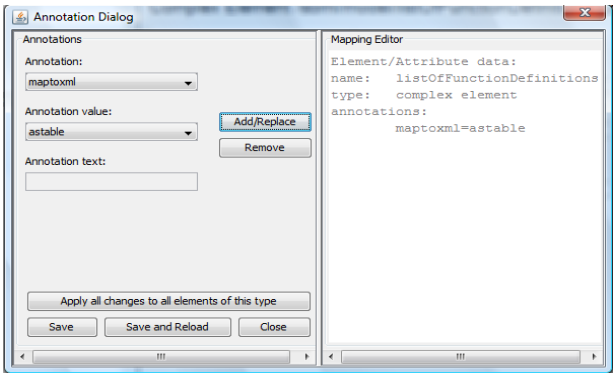

Figure 3. Add/remove annotations dialogue

in the schema tree when a particular dataset is loaded to the database in use. Three different colors are used to facilitate user's perception and to show how many times a particular child node appears under its parent element i.e., different children nodes are colored depending on their frequency of appearance. Thus, the user gets fast and highly useful overview of children nodes and can prioritize his next studies based on this information. The schema tree representation of statistical information aids the user decision on what annotations are appropriate to be used for a particular dataset and helps to construct proper queries with higher efficiency. Further, the statistics can help the user to create indexes and optimize queries.

The other part of the statistical data described in the previous section can be found in "Open Main Statistics" and "Open All Statistics" dialogs under the "File" menu. The statistical data visualized in the schema tree is small, however, our experience have shown that it is the most useful part of the information available for the dataset.

V. A FIRST EVALUATION

In this section, we present the results from the preliminary study of our approach, a more extensive research could be a subject of a future work. To explore the benefit of our tool and the statistical information, we used it to evaluate the performance on the homo sapiens dataset from the REACTOME database [18] and on the BIOMODELS dataset [7], both corresponding to the SBML 2.1 XML schema [14]. The data in the first dataset is spread in depth (the data is stored on many levels) and the data in the second dataset is spread in width (the data is populated almost equally within the dataset).

The statistical information for the dataset of interest becomes available in HShreX when it is loaded in the database in use. The statistics available directly in the HShreX schema tree gives detailed information for the occurrence of the nodes and their parents and present a clear view of data distribution in the particular dataset. This data is presented in the interface as a number pair in the child node name where the first number shows the number of times the child element occurs under its parent and the second shows the number of times the parent element is presented in the dataset at this level. The three numbers, in the parenthesis in

 ISBN:978-1-61208-115-1

Shredded:
```
SELECT a."id", a."name"
FROM sbml_model_listOfReactions_reaction a,
          sbml_model_listOfReactions_reaction_listOfReactants b,
          sbml_model_listOfReactions_reaction_listOfReactants_speciesReference c
WHERE a."shrex_id" = b."shrex_pid"
        AND b."shrex_id" = c."shrex_pid"
         AND c."species" = 'REACT_5251_1_Oxygen';
```

Native:
```
SELECT reaction.query(    'for $i in /reaction/listOfReactants/speciesReference
            where  $i/@species = "REACT_5251_1_Oxygen"
            return <Details> {$i/../../@id} {$i/../../@name} </Details>') "data"
FROM sbml_model_listOfReactions_reaction
WHERE reaction.exist('/reaction/listOfReactants/speciesReference
                          [@species="REACT_5251_1_Oxygen"]') = 1;
```

Figure 4. Sample query for SBML – Query 1

Shredded:
```
SELECT d."species", b."shrex_pid", e."species
FROM sbml_model_listOfReactions_reaction_listOfReactants b,
          sbml_model_listOfReactions_reaction_listOfProducts c,
          sbml_model_listOfReactions_reaction_listOfReactants_speciesReference d,
          sbml_model_listOfReactions_reaction_listOfProducts_speciesReference e
WHERE c."shrex_pid" = b."shrex_pid"
        AND b."shrex_id" = d."shrex_pid"
        AND c."shrex_id" = e."shrex_pid"
         AND d."species" = 'REACT_5251_1_Oxygen';
```

Native:
```
SELECT reaction.query(    'for $react in //reaction,
            $rtant in $react/listOfReactants/speciesReference,
            $prod in  $react/listOfProducts/speciesReference
            return <path> {data($rtant/@species)} {data($react/@id)}
                                {data($prod/@species)} </path>') "test"
FROM sbml_model_listOfReactions_reaction
WHERE reaction.exist('//reaction/listOfReactants/speciesReference
                          [@species="REACT_5251_1_Oxygen"]') = 1;
```

Figure 5. Sample query for SBML – Query 2

the child node name, show the minimum, the maximum and the average number of times this child occurs for this parent.

Examining the mentioned datasets, using the HShreX interface, we noticed that some of the elements and their parents occur more often than others, thus our research will be more productive if we concentrate on them. Therefore in our examples we have applied the HShreX annotation **maptoxml** to the *reaction* and to the *model* elements in the XML schema. This particular annotation/value combination has been selected in order to force the HShreX application to store these parts of the data as pure XML in the corresponding database. If we do not apply any HShreX annotations the data in the datasets is represented in a shredded storage model (positions 1 and 2 in Figure 6). The HShreX has been forced to represent the data in a hybrid and in a pure native storage models applying the **maptoxml** annotation to the *reaction* (positions 4 to 8) and *model* (positions 10 to 14) elements respectively.

We have chosen two of the major database servers available on the market and set up their options related to the XML data representation in various configurations. Using the database servers XML storage capabilities we are able to store the XML data with or without associating it with corresponding XML schema. The database servers run on HP Proliant DL380 G6 Server with two Intel Xeon E5530 Quad Core HT Enabled processors running at 2.4 GHz (in total 16 logical processors) and 30 GB RAM.

We have created different SQL queries (exemplified in Figure 4 and Figure 5) and executed them against the three storage models and different database configurations. In Query 1, the simpler among both, we retrieve details for a reaction where one of its participants is specified. In the second query, we join details for reactions and reactions to extract participants and products for all reactions. First we executed the two queries using only the homo sapiens dataset. After that we loaded both datasets at the same time and evaluated how the response time changes when the size of the data stored in the database increases. The measured performance can be influenced by other processes running on the server. To reduce this influence, the queries from the figures were executed ten times per condition set, and the averages of the results are presented.

First runs were made without any additional optimization. Based on the statistics, proper XML indices, for each variation of database storage options, were created and the same queries were executed again. Thus, we benefit from the statistical information available for a particular dataset in three ways: we can use the statistics to choose the best place for the HShreX annotations regarding our interests and in this way to switch flexibly and rapidly between different storage models. We are as well able to create proper, for each storage model, indices based on the view of the data distribution in the particular dataset. A final advantage is that we can optimize our SQL queries not only creating indices but rewriting them based on the data

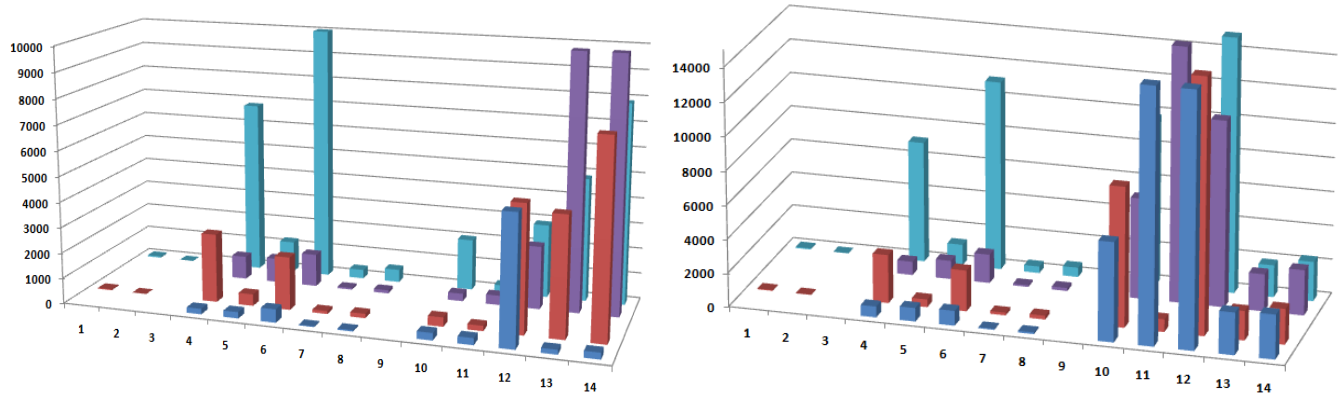

Figure 6. Performance [ms] for Query 1 (left) and Query 2 (right) where: ■ homo sapiens dataset with index, ■ homo sapiens dataset without index, ■ homo sapiens and biomodels datasets with index and ■ homo sapiens and biomodels datasets without index

distribution and complexity.

The results from the two different query executions are shown on Figure 6. The equivalent positions on the 'X' coordinate in both of the charts correspond to equivalent condition sets of database storage options. The results from positions 1 and 2 correspond to a fully shredded storage, positions 4 – 8 correspond to a hybrid storage and positions 10 – 14 correspond to a pure native XML storage. Positions 4 – 8 use the same conditions sets of database storage options as positions 10 – 14, however the HShreX annotation is applied to different elements. As we expected, there is a clear relation between the storage model and the query performance i.e., the execution times are fastest in the shredded storage and slowest in the pure native storage.

Examining the positions 4 – 14 in both result sets we can clearly see that the query performance varies with a different amount for the different database storage options when the size of the data in the database increases. The performance is usually improved when the XML indices are created. It is worth noting that this is not true for position 11 in Query 2 where the performance drops considerably when the index is used. While positions 4 – 8 in the two results sets are comparable, positions 10 – 14 have a lot of differences. Positions 13 and 14 in the first results set have the worst performance among the results for pure native storage while in the second results set they have the best performance. Analyzing positions 13 and 14 in the first result set shows that indices have excellent performance when the size of the data is relatively small and their performance decrease when the data size increases. It is worth noting as well the differences between positions 7, 8 and respectively 13, 14 in the results for Query 1. Positions 7, 13 and 8, 14 respectively have the same database storage options – positions 7 and 8 give the best results while positions 13 and 14 give the worst.

Analyzing the two result sets we can conclude that indices provide better results when used with the hybrid storage than with the pure XML storage. The indices efficiency increases when the size of the data in the hybrid storage increases. During results analysis we need to consider that the results are also affected from the database servers XML storage capabilities and created indices. The benchmark results are influenced from the data distribution in the datasets as well as the SQL queries construction. The statistical data available in HShreX facilitates and aids our decision where to put HShreX annotations and SQL indices and thus HShreX assists us in fast storage construction.

VI. Related Work

The work presented in this article combines ideas from several different areas for XML storage. The first is the work on automatic shredding of XML documents into relational databases by capturing the XML structure or based on the DTD or XML schema for the XML data [1][5][6][8]. The intention with these approaches is to create efficient storage for the XML data. The resulting data model is often hard to understand and is usually hidden from the user via an interface providing automatic query translation of XQuery into the model.

The other related area is hybrid XML storage for relational databases. The vendors offer different underlying representation for the XML type, in some cases it is a byte representation of the XML, in other cases it is some kind of shredding of the XML data [4][10][20][23]. In addition, database vendors provide a number of tools to import XML natively or shred the data into the system. These tools are intended for design of one database solution, thus generation and evaluation of alternative solutions become time consuming.

Interesting work [21] has addressed the question of properties of XML data and generating statistical and comparative measurements of XML datasets. However, this work concentrates on overall measures of properties of the dataset and does not consider the more detailed statistical measurements that we have found most useful in our work.

Other related works are found within database optimization [11][15]. Query optimization can rely on statistics of data and query use for fine tuning their performance [9][12]. However, these statistics are often dependent on the internal database representation instead of based on the original dataset as is necessary for our work. It would be interesting to include these measurements in our work to see whether they could give added value to our indicators.

VII. Conclusion and Future Development

The first tests of the tool are promising and show that our tool is very useful for aiding in storage design. Using the tools and statistics improves the evaluation process and makes it possible to compare a high number of alternative hybrid database designs. We will continue to use the tool for more extensive evaluations and to refine the method. In particular, we want to compare our set of measurements with the more advanced statistical methods used in [9]. The final goals would be to use the measure to provide suggestions of beneficial hybrid data models for the end user, to further automate the process of storage design. To reach this goal it is crucial to have access to series of data with specific properties to fine tune the indicators and tests. Also for this issue we have made a first solution for generating data with desired properties [13], which can be integrated into our tool.

One bottleneck with our method is that hybrid data models are very complex to query due to the mix of query languages. We are currently using SQL/XML, however, if we consider a user that want to work on the data as if it was XML, this is not feasible. Options are automatic query translations from XQuery to the defined model or to provide a higher level query language for the user.

Another very interesting question is hybrid storage solutions with several DB architectures as a backend, for instance pure native XML databases or specialized databases for graphs or RDF storage. This becomes particularly important for applications where parts of the data contain RDF code or represent graphs as is the case for many system biology standards. We have previously evaluated different combinations [27][28] and would like to include also these options in the HShreX Framework.

 ISBN:978-1-61208-115-1

ACKNOWLEDGMENT

We acknowledge the financial support from the Center for Industrial Information Technology and the Swedish Research Council. We are also grateful to Juliana Freire for support and fruitful discussions regarding this work and for Mikael Åsberg for implementation work on the HShreX tool.

REFERENCES

[1] S. Amer-Yahia, F. Du, and J. Freire, A Comprehensive Solution to the XML-to-Relational Mapping Problem, Proceedings of the ACM International Workshop on Web Information and Data Management, Nov. 2004, pp. 31-38, doi:10.1145/1031453.1031461.

[2] B. Aranda et al., The IntAct molecular interaction database in 2010, Nucleic Acids Research, Oct. 2009, pp. 1-7, doi:10.1093/nar/gkp878.

[3] D. Barbosa, J. Freire, and AO. Mendelzon, Designing Information-Preserving Mapping Schemes for XML, Proceedings of the International Conference on Very Large Databases, Aug.-Sep. 2005, pp. 109-120.

[4] K. Beyer, F. Özcan, S. Saiprasad, and B. Van der Linden, DB2/XML:Designing for Evolution, Proceedings of the ACM SIGMOD International conference on Management of data, Jun. 2005, pp. 948-952, doi:10.1145/1066157.1066299.

[5] B. Bohannon, J. Freire, P. Roy, and J. Siméon, From XML Schema to Relations: A Cost-Based Approach to XML Storage, Proceedings of the IEEE International Conference on Data Engineering, Feb.-Mar. 2002, pp. 64-75, doi:10.1109/ICDE.2002.994698.

[6] F. Du, S. Amer-Yahia, and J. Freire, ShreX: Managing XML Documents in Relational Databases, Proceedings of the International Conference on Very Large Databases, Aug.-Sep. 2004, pp. 1297-1300.

[7] European Bioinformatics institute http://www.ebi.ac.uk/biomodels-main/ 25.09.2010.

[8] D. Floresco and D. Kossmann, Storing and Querying XML Data using an RDMBS, IEEE Data Engineering Bulletin, vol. 22(3), 1999, pp. 27-34.

[9] J. Freire, JR. Haritsa, M. Ramanath, P.Roy, and J. Siméon, StatiX: making XML count, Proceedings of the ACM SIGMOD International conference on Management of data, Jun. 2002, pp. 181-191, doi:10.1145/564691.564713.

[10] H. Georgiadis and V. Vassalos, XPath on steroids: Exploiting relational engines for XPath performance, Proceedings of the ACM SIGMOD International conference on Management of data, Jun. 2007, pp. 317-328, doi:10.1145/1247480.1247517.

[11] G. Gottlob, C. Koch, and R. Pichler, Efficient Algorithms for processing Xpath Queries, ACM Transactions on Database Systems, vol. 30, No 2, Jun. 2005, pp. 444-491, doi:10.1145/1071610.1071614.

[12] T. Grust, J. Rittinger, and J. Teubner, Why Off-the-Shelf RDMBMSs are Better at Xpath Than You Might Expect, Proceedings of the ACM SIGMOD International conference on Management of data, Jun. 2007, pp. 949-958, doi:10.1145/1247480/1247591.

[13] D. Hall and L. Strömbäck, Generation of Synthetic XML for Evaluation of Hybrid XML Systems, In: M. Yoshikawa et al. (Eds) Database Systems for Advanced Applications 15th International Conference, International Workshops: GDM, BenchmarX, MCIS, SNSMW, DIEW, UDM, Apr. 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6193, 2010, pp. 191-202, doi:10.1007/978-3-642-14589-6_20.

[14] M. Hucka et al., The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models, Bioinformatics, vol. 19(4), 2003, pp. 524-531, doi:10.1093/bioinformatics/btg015.

[15] J. McHugh and J. Widom, Query optimization for XML, Proceedings of the International Conference on Very Large Databases, Sep. 1999, pp. 315-326.

[16] I. Mlynkova, Standing on the Shoulders of Ants: Towards More Efficient XML-to-Relational Mapping Strategies, Proceedings of the International Workshop on Database and Expert Systems Applications, Sep. 2008, pp. 279-283, doi:10.1109/DEXA.2008.16.

[17] MM. Moro, L. Lim, and Y-C. Chang, Schema Advisor for Hybrid Relational-XML DBMS, Proceedings of the ACM SIGMOD International conference on Management of data, Jun. 2007, pp. 959-970, doi:10.1145/1247480-1247592.

[18] Reactome – a curated knowledgebase of biological pathways http://reactome.org 25.09.2010.

[19] L. Runapongsa, JM. Patel, HV. Jagadish, Y. Chen, and S. Al-Khalifa, The Michigan Benchmark: Towards XML Query Performance Diagnostics, Information Systems, vol. 31(2), Apr. 2006, pp. 73-97, doi:10.1016/j.is.2004.09.004.

[20] M. Rys, XML and relational Management Systems: Inside Microsoft SQL Server 2005, Proceedings of the ACM SIGMOD International conference on Management of data, Jun. 2005, pp. 958-962, doi:10.1145/1066157.1066301.

[21] I. Sanz, M. Mesiti, G. Gurrini, and RB. Llavori, An entropy based characterization of the heterogeneity of XML collections, Proceedings of the International Workshop on Database and Expert Systems Applications, Sep. 2008, pp. 238-242, doi:10.1109/DEXA.2008.55.

[22] AR. Schmidt, F. Waas, M. Kersten, MJ. Carey, I. Manolescu, and R. Busse, XMark: A Benchmark for XML Data Management, Proceedings of the International Conference on Very Large Databases, Aug. 2002, pp. 974–985.

[23] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, and J. Naughton, Relational databases for querying XML documents: Limitations and opportunities, Proceedings of the International Conference on Very Large Databases, Sep. 1999, pp. 302-314.

[24] L. Strömbäck, Possibilities and Challenges Using XML Technology for Storage and Integration of Molecular Interactions, Proceedings of the International Workshop on Database and Expert Systems Applications, Aug. 2005, pp. 575-579, doi:10.1109/DEXA.2005.154.

[25] L. Strömbäck and J. Freire, XML Management for Bioinformatics Applications, Computing in Science and Enineering, in press.

[26] L. Strömbäck and D. Hall, An evaluation of the Use of XML for Representation, Querying, and Analysis of Molecular Interactions, In: T. Grust et. al. (Eds) Current Trends in Database Technology – International Conference on Extending Database Technology 2006 Workshops PhD, DataX, IIDB, IIHA, ICSNW, QLQP, PIM, PaRMA, and Reactivity on the Web, Mar. 2006, Revised Selected Papers. Lecture Notes in Computer Science, vol. 4254, 2006, pp. 220-233, doi:10.1007/11896548_20.

[27] L. Strömbäck, D. Hall, M. Åsberg, and S. Schmidt, Efficient XML data management for systems biology: Problems, tools and future vision, International Journal on Advances in Software, vol. 2(2-3), 2009, pp. 217-233, Invited contribution.

[28] L. Strömbäck and S. Schmidt, An Extension of XQuery for Graph Analysis of Biological Pathways, Proceedings of the International Conference on Advances in Databases, Knowledge, and Data Applications, Mar. 2009, pp. 22-27, doi:10.1109/DBKDA.2009.16.

[29] L. Strömbäck, M. Åsberg, and D. Hall, HShreX – A Tool for Design and Evaluation of Hybrid XML storage, Proceedings of the International Workshop on Database and Expert Systems Applications, Aug.-Sep. 2009, pp. 417-421, doi:10.1109/DEXA.2009.33.

[30] The UniProt Consortium The Universal Protein Resource (UniProt), Nucleic Acids Research, vol. 36(1), 2008, pp. D190-D195, doi:10.1093/nar/gkm895.

[31] BB. Yao, MT. Özsu, and N. Khandelwal, XBench Benchmark and Performance Testing of XML DBMSs, Proceedings of the IEEE International Conference on Data Engineering, Mar. 2004, pp. 621-633.