# Visual Querying and Exploration of Large Answers in XML Databases with $X^2$: A Demonstration

Holger Meuss
Center for Information and Language Processing
University of Munich, Germany
meuss@cis.uni-muenchen.de

Klaus U. Schulz
Center for Information and Language Processing
University of Munich, Germany

François Bry
Institute for Computer Science
University of Munich, Germany

## Abstract

*This paper describes $X^2$ (e***X***ploring* **X***ML data), a system for efficiently querying XML data and visually exploring answer sets, with a special focus on the novel user interface for interaction with the query result. The process of querying XML documents is performed interactively and in a completely graphical manner. Special attention was paid to simplicity of the user interface: In a first step the user submits a query to the system and gets the result presented in the form of a Complete Answer Aggregate (CAA). CAAs visually (re)present the answer space in a compact form and thus provide a good overview over the set of all answers. They offer a rich variety of presentation modes and exploration techniques which are used in the second step to interactively explore the answer space.*

## 1 System architecture and query language

The overall architecture of $X^2$ is depicted in Figure 1. The main components of $X^2$ are a query engine and the CAA engine. The query engine evaluates a query submitted by the user and computes the resulting CAA. It uses dedicated index structures that guarantee efficient evaluation. After query evaluation is completed, the resulting CAA can be explored by interaction with the CAA engine.

Since XML data is tree-structured data, we use a simple language for querying trees that extends Kilpeläinen's Tree Matching model [3]. The user draws a tree-shaped query (Figure 2) that will be matched with the tree-structured XML data. The user has the possibility to specify direct and indirect containment of elements, attribute values, text containment, element names, order and direct neighborhood of elements, and optional elements. The query language is described in more detail in [6].

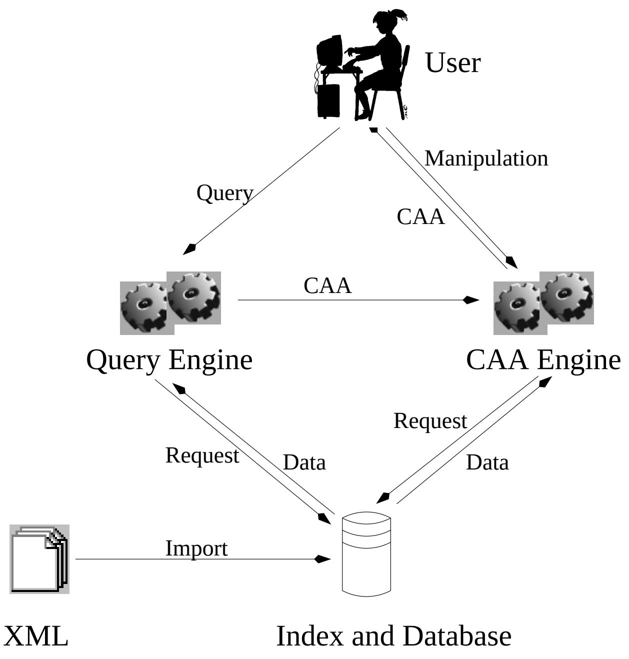

**Figure 1. System architecture of $X^2$**

## 2 Complete Answer Aggregates

A Complete Answer Aggregate (CAA), as depicted in Figure 3, is a datastructure that stores for every query node all matching database elements, together with their containment relations. Figure 3 shows an example CAA for a query searching for cities with their respective sightseeing infor-

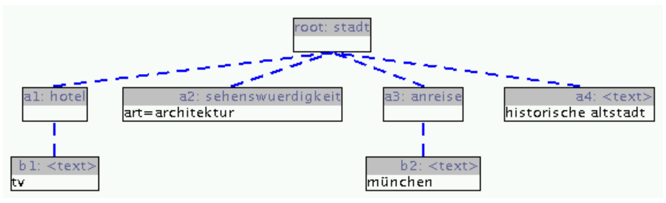

**Figure 2. $X^2$ query**

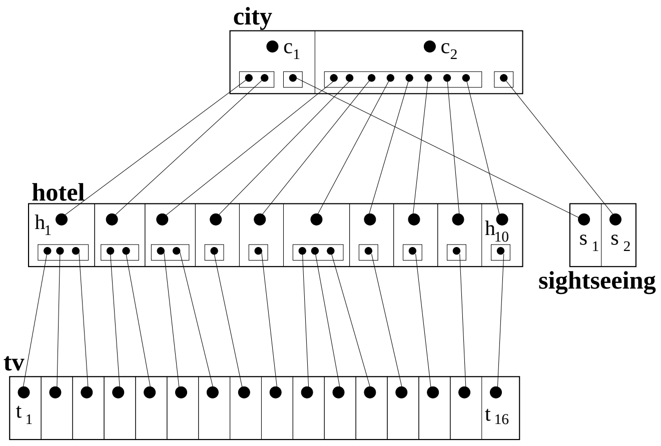

**Figure 3. Complete Answer Aggregate**

mation and hotels whose description contain the word “tv”. Two cities ($c_1, c_2$) have been found for this query, together with 2 sightseeing elements, and 10 hotels with 16 occurrences of the string “tv” in total. The first city contains two matching hotel elements, the second city eight.

Complete Answer Aggregates can be computed very efficiently due to their compact (polynomial-sized) representation of an exponential number of answers (cf. [6, 7]). Queries are evaluated pathwise with the help of a dedicated path index structure [5]. Evaluation is based on a dynamic programming algorithm that guarantees that every database node is touched at most a constant number of times.

## 3 Exploring Complete Answer Aggregates

The design of the user interface of $X^2$ was guided by the observation that the user needs help in localizing the most useful pieces of information in large answer sets, especially in a complex information space as often constituted by XML data. We will describe the main characteristics of the user interface by an example that uses an existing XML database describing cities in Bavaria. Each city, represented by an XML document, has a rich description: name, sightseeing places, hotels, conference sites, restaurants, regular events, etc. In our scenario a user wants to retrieve information about cities (“stadt”) with a historical town center (“historische altstadt”), hotels with a tv set, sightseeing places (“sehenswuerdigkeit”), and transportation facilities (“anreise”) from Munich to these cities. This information need is represented by the query in Figure 2.

A CAA is presented in a first view only as the query itself where each query node is tagged with the number of matching database nodes (as can be seen in the presentation mode of query node “anreise” in Figure 4). This simple overview serves as a starting point for further exploration. The user may now request to see all matching database nodes for a query node using a context menu associated with that query node. Every database node is then displayed with a short summary generated out of the contained text (see query node “stadt” in Figure 4). The user may also switch to a detailed summary (see query nodes “hotel” and “sehenswuerdigkeit” in Figure 4). These summaries are generated according to the respective element types: Hotel and city elements, e.g., have name and description subelements which are used for generating the summary. Different summary generation algorithms can be plugged into $X^2$: Returning the first characters of the element content is the default, if no algorithm is given for an element type. It is also possible to view the full text of a document and display directly the textual content of a given element. The document text is displayed formatted (using XSLT stylesheets) and the search terms (“tv”, “münchen”, “historische altstadt” in the example) of the query are highlighted.

The user may hide database and query nodes in order to improve on partial overview. It is also possible to mark database nodes and to restrict the screen presentation to nodes related to marked database nodes: The user might decide that one sightseeing place (element “sehenswuerdigkeit”) is especially interesting and mark it. He might then declare that only database nodes within the same substructure (i.e. city) shall be displayed. It is now easier for the user to explore, for example, the hotels of the city in question without being distracted by hotels of other cities.

The user may also sort database nodes according to various criteria, for example on attribute values. Continuing the example, the user can now sort the hotels with respect to their quality attribute value, with the 5 star hotels being displayed first. Finally, it is possible to group elements with respect to these criteria, so that all 5 star hotels are displayed in one group, all 4 star hotels in the next group, and so on.

Of, course, all these exploration techniques can be applied in a cumulative way, e.g. grouping the visible hotels only after hiding hotels of a given city.

## 4 Implementation and Related Work

This paper presented $X^2$ with a focus on the graphical user interface for exploring the answer set to a query. $X^2$ has been fully implemented in Java and uses the Postgres DBMS for storing the index. $X^2$ has been tested with three

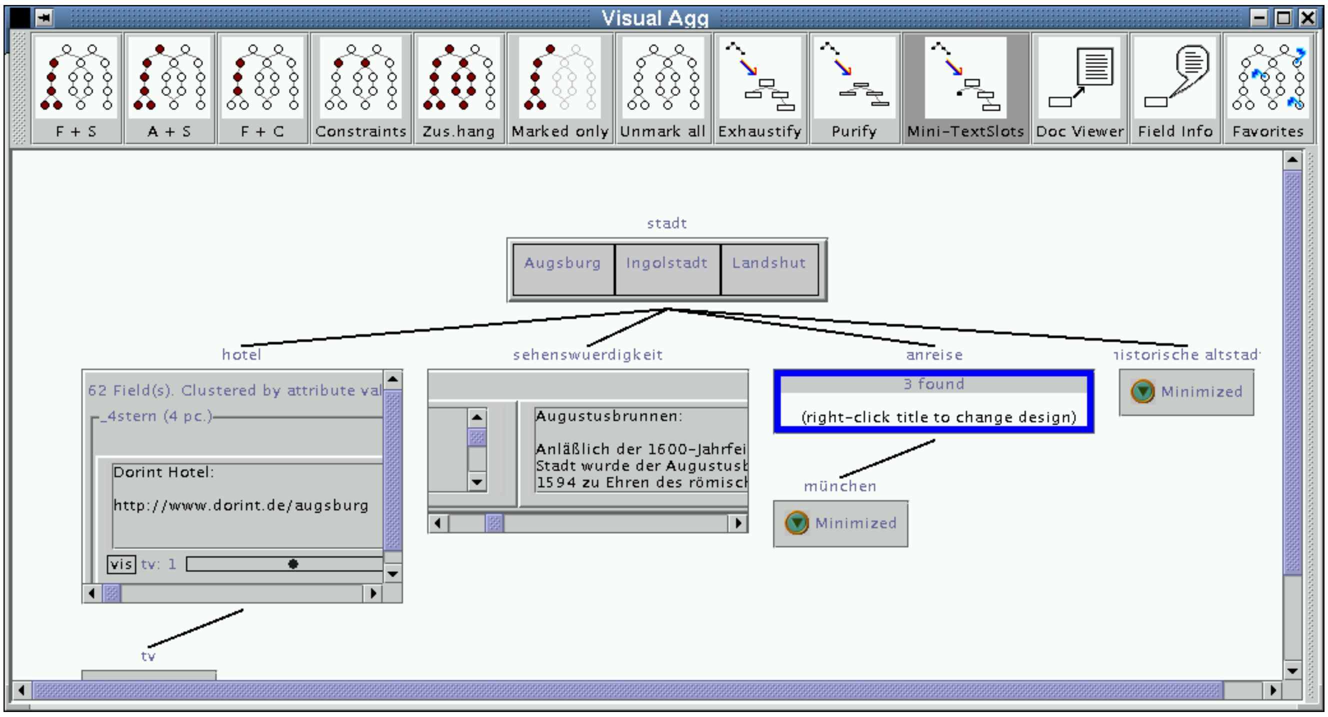

**Figure 4.** $X^2$ **user interface**

different document collections so far, ranging from 2MB to 500MB in size. For these collections, queries can be evaluated efficiently (less than 1 second). We still plan to improve the implementation of the graphical presentation, since the GUI slows down if the computed CAA contains too many database nodes.

So far, there has been limited research in graphical query languages for XML and user support in XML data exploration. XML-GL [1] is a fully graphical query language for retrieval and manipulation of XML data. But, in contrast to $X^2$, XML-GL does not support interactive and graphical exploration of XML data. DataGuides [2], a user interface for browsing XML data used in the Lore system [4], presents the user a DataGuide as a compact description of (parts of) the XML data. Indeed, CAAs are somehow similar to DataGuides, but a CAA's structure is inherited from both the user query and the database, whereas DataGuides are ignorant of query structure and owe their structure to the XML data only. BBQ [8] is a system with strong support for interactive query formulation, but with no support for result inspection.

**Acknowledgements** The authors would like to thank Christian Strohmaier for proofreading and Marcus Schilling for his implementation work on $X^2$.

## References

[1] S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: a graphical language for querying and restructuring XML documents. *Computer Networks*, 31(11–16):1171–1187, May 1999.

[2] R. Goldman and J. Widom. Interactive query and search in semistructured databases. In *WebDB'98, Proc. Int. Workshop on the Web and Databases*, 1998.

[3] P. Kilpeläinen. *Tree Matching Problems with Applications to Structured Text Databases.* PhD thesis, Dept. of Computer Science, University of Helsinki, 1992.

[4] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. *SIGMOD Record*, 26(3), 1997.

[5] H. Meuss. *Logical Tree Matching with Complete Answer Aggregates for Retrieving Structured Documents.* PhD thesis, Dept. of Computer Science, University of Munich, 2000.

[6] H. Meuss and K. U. Schulz. Complete answer aggregates for tree-like databases: A novel approach to combine querying and navigation. *ACM Transactions on Information Systems*, 19(2):161–215, April 2001.

[7] H. Meuss, K. U. Schulz, and F. Bry. Towards aggregated answers for semistructured data. In *ICDT'01, Int. Conf. on DB Theory*. Springer, 2001.

[8] K. D. Munroe and Y. Papakonstantinou. BBQ: A visual interface for browsing and querying of XML. In *5th IFIP Working Conf. on Visual Database Systems*, pages 277–296, 2000.