Interactive Query Formulation in Semistructured Databases

Agathoniki Trigoni

Athens University of Economics and Business
Department of Informatics
ntrig@nbg.gr

Abstract. The use of large amounts of distributed and heterogeneous information has become extremely cumbersome; this difficulty is mainly related to exploring the data, rather than actually storing or exchanging it. The user who is interested in small bits of information is getting more and more confused when having to dig under a large volume of diverse and more importantly semi-structured data. In this paper, we propose an interactive and adaptive framework that guides the user in the search for data, by disclosing only a part of the underlying information at a time. It first provides the user with a high-level view of the raw data and gradually adapts to his/her needs in order to offer a refined answer. The proposed model offers the possibility to query a semistructured database based on general schema-related constraints imposed by the user or identified by the system, but without specific knowledge of the underlying metadata. This is achieved by receiving initially an *amorphic* query, which may consist of one or more basic paths, and helping the user to refine it gradually to a specific semistructured query, expressed in a language like XQuery or Lorel.

1 Introduction

The queries processed in the areas of databases and information retrieval are tailored for the needs of the underlying systems. Regarding databases, query constructs vary depending on whether the context is an object-oriented [1], relational [2], deductive [3] or semistructured database [4–6]. In general, prior knowledge of the database structure is required, even if it is limited or is inferred automatically, as in the case of semistructured databases [7–9]. On the other hand, searches in the context of information retrieval have a different style; they are usually based on identifying a set of keywords in a series of documents, and using a variety of criteria and ranking algorithms to sort the resulting documents based on their relevance to the keywords [10].

Neither of the two query styles provides the most natural way of searching. The query framework proposed in this paper takes into account the following points:

- Users can usually contribute to the query input in a more intelligent way than by providing a few keywords.
- A system is not user-friendly when it requires a detailed knowledge of the database structure, especially in the presence of large amounts of heterogeneous data.

– The user would not like to miss information because of his lack of knowledge about the schema. Query languages in semistructured systems currently allow the user to avoid type errors when not complying with the explicit or implicit schema. However, the user ends up receiving only a part of the information requested.
– Users usually have a high-level knowledge of the database structure, based on their natural perception of data correlations.
– The lack of knowledge about the database structure does not preclude the need to enforce specific selection criteria on the results.

Structured queries require knowledge of schema information, or else fail to deliver complete and accurate results. IR-style searches do not exploit but a small part of the user's knowledge about the data (mainly keywords), and result in sets of documents that cannot be filtered using detailed criteria. The idea behind this paper is that the tradeoff between allowing *natural* queries and receiving quality (complete and accurate) results, could be compromised if we adopted an interactive and adaptive query model.

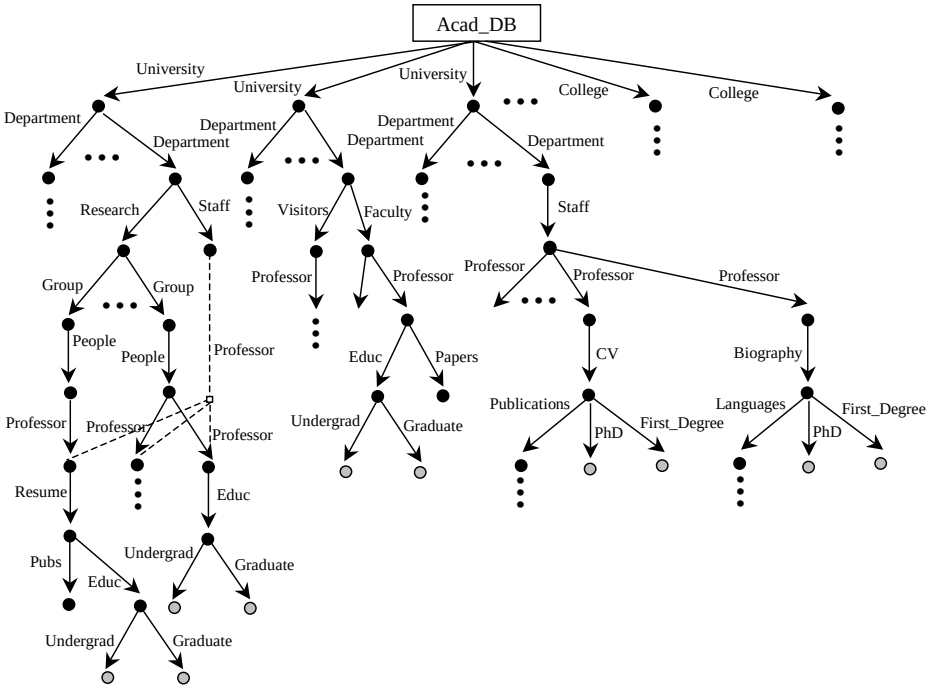

Fig. 1. XML data: Academic information about universities and colleges

Consider for instance an XML tree that represents academic information about universities and colleges (figure 1). The root of the tree named `Acad_DB` has several `University` labels and each one of them is organised in a different way. Say that a Scholarships Foundation is interested in funding the Computer Science department of a

university with strong background in mathematics. Hence, they look for the department with the maximum number of academics having studied maths in their first degree and computer science in their graduate studies. That is, they search for quite specific information without having specific knowledge of the data organisation, but merely a high level view of how universities are organised in general.

Using an IR approach and providing keywords like *maths*, *professor*, *informatics* they might be able to receive relevant Web documents, but still it would be difficult to identify automatically the department that satisfies their selection criterion. Since there is a need for an exhaustive search in all universities, well-known semistructured queries cannot be trusted, since they might exclude interesting information from the result, without even sending an error message, e.g. type conflicts. Our objective is the tight coupling of knowledge discovery and query processing, as a means of providing the user with the best answer possible, exploiting existing high-level knowledge of the domain context but also enabling low-level filtering techniques.

In what follows, we present the functionality of the proposed model. In section 2 we discuss the kinds of schema and semantic information used by the interactive query processor. This information is either discovered in advance or mined upon the request of a user. The algorithm that guides the interactive process is proposed in section 3. A concrete example that demonstrates the process of gradual query refinement is presented in section 4. Based on the example, we discuss the advantages of our framework compared to more traditional query or IR engines, and show how it could potentially be extended in order to take into account new kinds of knowledge about the data (section 5). Finally, section 6 refers to previous work related to our framework and section 7 presents concluding remarks highlighting the novel points of the proposed model.

2 Data Organization: Schema, Morphology and Semantics

In most existing database systems, queries, which are expressed in declarative languages, provide information about the location of the data requested, and include operations like filtering, grouping and ordering. In our model, it is not necessary to specify explicitly the location of data; instead a number of new search criteria can be used that are metadata-related and allow for the expression of a wider variety of queries. Prior knowledge of metadata is not required, but metadata information can be used as a requirement (or filter) for the expected results. Metadata information is also used by the query processor, e.g. in order to justify that the user requirements are not satisfiable, as it will be shown in section 4.

The kinds of metadata information used either as search criteria (by the user), or as guides for query refinement (by the query processor), are presented below. Similar information has also been used for the semantic optimization of semistructured queries [11, 12]. In what follows, paths are denoted as $\mathtt{p}$, $\mathtt{p_i}$ or $\mathtt{p}'_\mathtt{i}$, and represent either specific paths (e.g. `Student.course.name`) or path patterns (e.g. `Student.course.`$*$, `Student.`$*(2,4)$`.name`). The symbol $*$ represents a sequence of zero or more labels, whilst $*(\mathtt{m},\mathtt{n})$ represents a sequence of $\mathtt{m}$ to $\mathtt{n}$ labels. The notation $\mathtt{lab_1}|\mathtt{lab_2}$ denotes that either of the two labels can be used; for example `Person.name`$|$`job` includes paths `Person.name` and `Person.job`. More formally, a path (or path pattern) is a sequence of

zero or more elements of the form:

$$\begin{aligned}\texttt{element} ::= & \ \texttt{label} \\ | & \ \texttt{label}_1 | \ldots | \texttt{label}_k \\ | & \ * \\ | & \ *(\texttt{m}, \texttt{n})\end{aligned}$$

where $\texttt{label}, \texttt{label}_1, \ldots, \texttt{label}_k$ are names of labels in a semistructured database.

The expression $\texttt{p} \subseteq \texttt{p}'$ expresses that path $\texttt{p}$ satisfies path $\texttt{p}'$, i.e. all specific paths (sequences of labels) that are instances of $\texttt{p}$ are also instances of $\texttt{p}'$.

1. *Sequence patterns*. Consider for instance a rule of the form

   $$\forall \texttt{p}_1, \texttt{p}_2.\ \texttt{p}_1 \subseteq \texttt{p}_1' \wedge \texttt{p}_1.\texttt{p}_2 \Rightarrow \texttt{p}_2 \subseteq \texttt{p}_2'$$

   This rule expresses that if a path $\texttt{p}_1$ satisfies the pattern $\texttt{p}_1'$ and $\texttt{p}_1$ is followed by $\texttt{p}_2$, then the latter satisfies the pattern $\texttt{p}_2'$.
   This metadata information could be used by the user to search for all paths $\texttt{p}_1$ that satisfy the rule above. For instance, a user might be interested in all *authors* that have written only *fiction* and hence there are no other labels (e.g. *novel*, *essay*) that start from these author nodes.
   The rule above might also be useful to the query processor to inform the user about the sequence patterns of a path $\texttt{p}_1''$, such that $\texttt{p}_1'' \subseteq \texttt{p}_1'$ or to justify why a query $\texttt{p}_1''.\texttt{p}_2''$, such that $\texttt{p}_1'' \subseteq \texttt{p}_1'$ and $\texttt{not}(\texttt{p}_2'' \subseteq \texttt{p}_2')$ would yield no results.
2. *Sibling patterns*. Paths in semistructured databases often exhibit properties that are related to the properties of sibling paths, i.e. of paths that start from a common parent node. Consider for instance a rule of the form

   $$\forall \texttt{p}_1, \texttt{p}_2.\ \texttt{p}_1 \subseteq \texttt{p}_1' \wedge \texttt{p}_2 \subseteq \texttt{p}_2' \wedge \texttt{p}_1.\texttt{p}_2 \Rightarrow \exists \texttt{p}_3.\ \texttt{p}_3 \subseteq \texttt{p}_3' \wedge \texttt{p}_1.\texttt{p}_3$$

   That is, if path $\texttt{p}_1$ is followed by $\texttt{p}_2$ then it is also followed by at least one path $\texttt{p}_3$, where $\texttt{p}_1 \subseteq \texttt{p}_1'$, $\texttt{p}_2 \subseteq \texttt{p}_2'$ and $\texttt{p}_3 \subseteq \texttt{p}_3'$.
   The query processor could use this semantic information in order to present the properties of a given path $\texttt{p}_1$ or $\texttt{p}_1.\texttt{p}_2$ and thus help the user to refine it to a specific query.
   The rule itself could be presented by the user as a query, e.g. find all paths $\texttt{p}_1$ that satisfy (or not) the rule. This query could probably be expressed using existing query languages for semistructured databases, but if the consequent of the rule was a universal quantification, this would be impossible.
3. *Shortcut properties* Consider for instance the rule

   $$\begin{aligned}&\forall \texttt{p}_1, \texttt{p}_2.\ \texttt{p}_1 \subseteq \texttt{p}_1' \wedge \texttt{p}_2 \subseteq \texttt{p}_2' \wedge \texttt{p}_1.\texttt{p}_2 \Rightarrow \\ &\exists \texttt{p}_3.\ \texttt{p}_3 \subseteq \texttt{p}_3' \wedge \texttt{end\_node}(\texttt{p}_1.\texttt{p}_2) = \texttt{end\_node}(\texttt{p1.p3})\end{aligned}$$

   The rule expresses that a path $\texttt{p}_1.\texttt{p}_2$ leads to the same point as at least one alternative path $\texttt{p}_1.\texttt{p}_3$. If $\texttt{p}_3$ is shorter (has fewer labels) than $\texttt{p}_2$ then the former is a shortcut of the latter.
   The query processor might propose the use of a shorter path than the one specified initially, in order to reveal potentially interesting schema information and thus guide

the user to express a more appropriate query. Shorter paths might also contribute to the more efficient execution of the initial query.
On the other hand, the user might be interested in paths that satisfy the pattern $p'_1.p'_2$ and that have at least one alternative path. For instance, it might be interesting to search for all countries such that if they export products to a certain country, they also import other products from it.

4. *Specificity* Consider for instance the following rule:

$$\forall p.\, p \subseteq p' \Rightarrow p \subseteq p''$$

It expresses the metadata information that paths that satisfy the pattern p' also satisfy the more specific pattern p''. If such a rule is true in the database, then the query of a user that involves a path $p_1 \subseteq p'$ should become more specific, say p_2 so that $p_2 \subseteq p''$. Exploiting metadata information, the query processor helps the user to refine his/her queries. Notice that this information is only exploited by the query processor and is not usually expressed as a query by the user.

5. *Topological properties* Topological information refers to the relevant location of a path, e.g. to its distance from a certain node. For instance the semantic information $\texttt{distance}(\texttt{anchor\_node}, p)$ $\texttt{in}$ R expresses the fact that the number of labels leading from $\texttt{anchor\_node}$ to the beginning of path p are in range R, e.g. R=[2,4]. The information $\texttt{bottom\_distance}(p)$ $\texttt{in}$ R denotes that the distance of the end of path p to a leaf (to which it leads) is in range R.
A user might be interested in paths that are close to leaf nodes, since leaves contain values upon which one may apply selection criteria. For instance, a user might query all paths p such that $\texttt{bottom\_distance}(p)$ $\texttt{in}$ $[1, 3]$.
If a user has very little knowledge of the structure of an XML tree, he might be interested in discovering gradually labels that are in the initial levels, and hence denote general ideas, or in labels close to leaves, which express what kind of values are stored in the database.

6. *Cardinality and Length of patterns*
Users are often interested in paths that occur a certain number of times in the database, independent of their location. The expression $\texttt{number}(p)$ $\texttt{in}$ R denotes that the cardinality of paths of the form p are in range R. For instance, in a survey about labour conditions in U.K. it is expected that labels that link individuals with jobs should be almost as many as the people of the sample. Hence, even if we have no prior knowledge of the specific label, its cardinality can be used as a starting point to enter and further explore the database. The problem of finding frequent subsequence patterns of the form $*.X_1.*.X_2.*...*.Xn.*$, where X_i is a subsequence of at least ℓ_i labels ($i = 1, \ldots, n$), has been studied in the context of combinatorial pattern discovery for scientific data [13]. In that context, the notion of maximum edit distance (as a dissimilarity measure) is used so that the support of a pattern is also increased by sequences that approximately satisfy a pattern (within edit distance d). This dissimilarity measure can also be used in our work, in order to allow a user to express queries of the form: "Find all subpaths of length 3 (i.e. $lab_1.lab_2.lab_3$) that occur at least 100 times in the database within distance 1".
The length of paths satisfying a pattern can be used in order to avoid cyclic paths, or indirect paths irrelevant to the user requirements. For instance, in a genealogic

XML graph, it is sufficient to use paths of length at most 3 in order to find the close relatives of a certain person. If a user specifies a query pattern satisfiable by many paths of completely different lengths, the query processor should notify the user of this information, in order to help him/her refine the query.

7. *Pattern Similarity* In our discussion about pattern cardinality (or support), we mentioned the problem of searching for frequent subsequences of labels of length ℓ within edit distance d. The user searches for frequent patterns of a minimum length, allowing for a certain degree of sequence dissimilarity. The dissimilarity measure can also be used in a similar problem, in which the user searches for all paths that contain a subsequence similar to an input path pattern. This need may occur for two reasons: i) a user might be unable to find any paths in the database satisfying the original pattern and ii) even if a pattern is found in the database, the user would like to cover the possibility of ignoring very similar and equally interesting patterns. Hence, given an initial path pattern, our framework provides the opportunity of searching the semistructured data for all similar paths within distance d, i.e. that satisfy the pattern if we modify (insert, delete or substitute) at most d of their labels.

It is interesting to notice that the kinds of metadata information defined above could be combined in several ways. For instance, a user might be interested in paths from a source to a target node, such that the number of alternative paths of length ℓ that link the two nodes are more than n. Likewise, the system could combine this knowledge in order to justify why a certain query is expected to yield no results, or as a way to aid in the refinement of the query. In the following section, we show the interactive process that exploits the metadata information defined above, namely path sequences, inheritance properties, connectedness and shortcuts, specificity rules, topological requirements, cardinality and length properties, as well as pattern similarity.

3 Interactive Search Algorithm

The idea behind the interactive search algorithm is that the user cannot express initially a specific query, but gradually refines a general one based on either metadata requirements that s/he enforces or metadata information about the schema that the query processor reveals.

The detailed algorithm is given below:

Step 1: The user initially presents a query, as a set of path templates (patterns).
Step 2: Repeat substeps 2.1 and 2.2

- *Substep 2.1:* The system uses metadata information about sequence and inheritance patterns, or specificity rules, in order to justify the absence of some input patterns, as well as to refine others. If metadata information relevant to the patterns is not available, the query processor navigates over the semistructured database in order to identify new knowledge and store it in the warehouse.
- *Substep 2.2:* The user adds, modifies, deletes, or accepts the resulting path templates.

Until the resulting patterns from substep 2.1 are not altered at all by the user in substep 2.2.

Step 3: Given the refined patterns, the user may proceed in either of the following ways:

1. *By looking for metadata information relevant to one or more of the patterns.*
   The user may select one or more of the following metadata information:
   - Sequence patterns
   - Sibling patterns
   - Connectedness and shortcut properties
   - Topology information
   - Cardinality and Length information
   - Similar patterns within edit distance `d`

   Based on the resulting metadata information, the user may modify some of the existing paths, or add new ones. The algorithm is then continued from step 2.
2. *By enforcing metadata conditions as requirements in order to further refine one or more patterns.*
   The requirements might concern one or more of the following metadata conditions:
   - Sequence patterns
   - Sibling patterns
   - Connectedness and shortcut properties
   - Topology information
   - Cardinality and Length information

   The metadata conditions result in more refined patterns, that the user might then accept, modify, reject or add new ones. The algorithm is then continued from step 2.
3. *By asking the system to present the user with all specific paths in the database that actually satisfy one or more of the refined paths.* In fact, the user may only be interested in the enumeration of ℓ consecutive labels preceding or following a segment (subpath) of a given pattern.
   In this case the user may select specific paths, or add, change, or delete some of the initial path patterns. The algorithm is then continued from step 2.
4. *If the patterns are already specific, the interactive process ends and the user may express a detailed query in a language like Lorel or XQuery.*

The functionality of the algorithm presented above is demonstrated through the use of an example in section 4.

4 Searching and Querying a Semistructured Database: Example

Consider the database consisting of academic information about universities and colleges (figure 1), which was first mentioned in section 1. The user is interested in identifying the university department with the maximum number of academics having studied maths in their first degree and computer science in their graduate studies.
Step 1

- *User:* The user registers interest for paths of the form:

$$P_1 : *.\texttt{University}.*.\texttt{Professor}.*.\texttt{Educ}$$

Step 2

– *System:* The system applies the specificity rule that all paths should start with the only entry point to the database `Acad_DB` and converts P_1 to P_2.

$$P_2 : \texttt{Acad\_DB.University.*.Professor.*.Educ}$$

It also uses the sequence pattern that all children of `University` labels are named `Department` to convert P_2 to P_3.

$$P_3 : \texttt{Acad\_DB.University.Department.*.Professor.*.Educ}$$

Step 3

– *User:* The user queries the length of subpaths `Department.*.Professor` in P_3 since he is interested in closely linked `Department` and `Professor` nodes. A long path between a `Department` and a `Professor` would signify that the latter does not belong to the former, but is rather remotely associated with it. If such a long subpath leads to the same `Professor` as a shorter path, then the user would be interested in considering only the shortcut.
– *System:* The answer is that we have paths of length 3 and 5.
– *User:* The user wonders why the subpaths `Department.*.Professor` are of two different lengths, and attempts to investigate the properties of the longer ones. If `Professor` nodes that are reached by paths of length 5 were also reached by paths of length 3, then the former paths could be ignored. Thus the user puts the condition that the length of `Department.*.Professor` in P_3 should be 5 and queries any connectedness properties of the resulting paths.
– *System:* The resulting subpaths of length 5 have the form P_4:

$$P_4 : \texttt{Department.Research.Group.People.Professor}$$

Regarding connectedness properties of P_4, all nodes reached by

$$\texttt{Department.Research.Group.People.Professor}$$

are also reached by

$$\texttt{Department.Staff.Professor}$$

Step 2

– *User:* Hence the user decides to refine P_3 by specifying that only one label separates `Department` and `Professor`:

$$P_5 : \texttt{Acad\_DB.University.Department.*}(1,1)\texttt{.Professor.*.Educ}$$

– *System:* No refinement is possible.

Step 3

– *User:* An enumeration of subpaths `Department.*`$(1,1)$`.Professor` (of path P_5) is requested.
– *System:* The answer is:

$$\texttt{Department.Faculty}|\texttt{Staff}|\texttt{Visitors.Professor}$$

Step 2:

– *User:* The user is not interested in `Visitors`, so s/he refines pattern P_5 to P_6. The user also adds the new pattern P_7 taking into account that education information about a professor might not necessarily follow an `Educ` node. Other relevant keywords could be `CV`, `Biography` or `Curriculum_Vitae`.

P_6 : Acad_DB.University.Department.Faculty|Staff.Professor.*.Educ
P_7 : University.*.CV|Biography|Curriculum_Vitae

– *System:* The system refines P_6 and P_7 to P_8 and P_9 respectively. Notice that label `Curriculum_Vitae` is not included in P_9.

P_8 : Acad_DB.University.Department.Faculty|Staff.Professor.*.Educ
P_9 : Acad_DB.University.Department.*.Professor.CV|Biography

Step 3

– *User:* The user searches for sibling patterns in P_8 and P_9. The reason is that s/he suspects that graduate and undergraduate studies (independent of their actual labels) are likely to occur together under either `Educ` or `CV|Biography`.
– *System:* A sibling pattern was identified in P_8:

$$\forall p.p \subseteq \texttt{Educ} \wedge p.\texttt{Undergrad} \Rightarrow p.\texttt{Graduate}$$

– *User:* Based on previous knowledge, the user refines path P_9 by replacing its subpath `Department.*.Professor` with `Department.Faculty|Staff.Professor`. Having no further specific requirements, s/he then asks for an enumeration of all labels that follow `CV` or `Biography` in P_9.
– *System:* `PhD|First_Degree|Publications|Languages`

Step 2 New paths presented by user.

P_{10} : Acad_DB.University.Department.Faculty|Staff.Professor.*.Educ.Undergrad
P_{11} : Acad_DB.University.Department.Faculty|Staff.Professor.*.Educ.Graduate
P_{12} : Acad_DB.University.Department.Faculty|Staff.Professor.CV|Biography.First_Degree
P_{13} : Acad_DB.University.Department.Faculty|Staff.Professor.CV|Biography.PhD

Step 3: The user queries the distance of these paths from leaves and the system confirms that it is 0.

Hence the user may use these paths to form the following query:

```
Q1 = (select dept : x.dept, no_of_prof : count(x.prof)
  from
 (select dept : Acad_DB.University.Department, prof : p
 from Acad_DB.University.Department.Faculty|Staff.Professor as p
 where p.*.Educ.Undergrad contains Maths
 and p.*.Educ.Graduate contains Computer|Inform
 union
 select dept : Acad_DB.University.Department, prof : p
 from Acad_DB.University.Department.Faculty|Staff.Professor as p
 where p.CV|Biography.First_Degree.university contains Math
 and p.CV|Biography.PhD contains Computer|Inform)
    as x
  group by dept : x.dept)
```

Q_1 returns all departments paired with the number of professors that have studied mathematics in their first degree and computer science or information techonology in their graduate studies. The following query selects the department with the maximum number of these professors.

```
select dept : y.dept
from Q1 as y
where y.no_of_prof =
  max(select z.no_of_prof from Q1 as z)
```

5 Advantages of the Interactive Search Algorithm

The example given above demonstrates a number of advantages of the proposed interactive search algorithm to traditional querying or IR techniques.

1. The user does not usually have a prior knowledge of schema information. Even if this information is revealed, it might be very confusing for the user to assimilate and handle. Our model provides the mechanism to gradually acquire relevant schema information to his/her information needs.
2. The user is enabled to do a search not only based on content criteria (keywords), neither only in exact location criteria (traditional queries). The model allows us to use more meaningful schema-related criteria, e.g. sequence, sibling, topological, connectedness, length and cardinality path constraints.
   - By looking for schema-related patterns, the user can gradually find his way to the paths that are relevant to his/her search. It is often more helpful to find our way through a graph, if we know general structure characteristics, rather than a detailed account of its paths.
   - By enforcing schema-related constraints, the user manages to limit the scope of the database, keeping a view only to the parts that satisfy the constraints. This is a powerful mechanism for users who do not know the exact location of the data, but have some intuitive knowledge of the constraints satisfied by the interesting paths. Since these constraints are structure- rather than value-related, they cannot easily be expressed in the *where* clause of a *select* query. However, they are very useful in refining the paths presented initially by the user.
   - As a last resort the user may request an enumeration of all possible paths satisfying a certain pattern. Such an enumeration is useful, because it is done in a local context (after paths are already refined and the scope of the database is limited). If it was performed in order to reveal the organisation of a whole semistructured database, the amount of information would be overwhelming and rather useless.
3. The iterative nature of the algorithm allows the user to readapt his search based on the additional knowledge acquired in previous steps. The user may interrupt the iterative process, only when s/he is satisfied with the expected degree of completeness of the result.

4. The interactive algorithm is extendible in two ways: First, new kinds of patterns or constraints could be searched or enforced respectively, as long as they are supported in programming terms. For instance, distributed, or mobile data sources could be queried based on semantic information about the distribution or movement patterns. Second, ontological information could enhance the functionality of the system significantly [14–16]. For instance, based on labels defined by the user, the system may reveal synonyms or relevant concepts, that could be parts of equally interesting paths. Ontologies have been traditionally used to enhance IR searches; however, their use should also be beneficial in the process of defining more conventional *select* queries.
5. Our model is a hybrid system that combines characteristics of both IR and conventional query models. The user may start with a few keywords (as in IR) and by gradually identifying or imposing structural database constraints, s/he may then refine the search gradually, until a detailed query is formed. Hence, the user may satisfy his need for applying detailed selection criteria, despite the initial lack of specific knowledge about the database schema. This is a very common need, and hence our model is expected to be useful for a variety of applications.

6 Related Work

The World Wide Web presents a great number of challenges from the viewpoint of database theory [17], including storing and querying data, imposing constraints and mining patterns [18]. As is stated in [17], 'a database is a polished artifact', compared to 'a free-evolving, everchanging' collection of data sources constituting the Web. Hence, there is a need to devise flexible models for finding and extracting knowledge either by querying raw XML-ized data or by mining patterns from semistructured databases or their corresponding schemata (DTDs).

Several languages have recently been proposed for querying XML databases [4, 5, 19, 6]. On the other hand, many research groups have focused on integrity or typing constraints applying to the semistructured model [20, 21]. These constraints have either been used for query optimization or as a means of preventing anomalies during updates or data integration. However, very little work has focused on a flexible mechanism for querying information with the aid of constraints. Amongst the few tools available for helping the user query a semistructured database, it is worth mentioning Document Type Descriptors (DTDs). A DTD, which may optionally accompany an XML document, serves the role of a schema specifying the internal structure of the document. A significant amount of work has studied the inference of a DTD from one or more documents, as well as the problem of deciding whether a document conforms to a given DTD [7–9]. XTRACT, a recent system for inferring a DTD schema, applies the Minimum Description Length (MDL) principle in order to generate both concise and intuitive DTDs containing regular expressions. Semantically meaningful DTDs can be used as a starting point in order to identify constraints, e.g. sequence or sibling patterns, using a variety of data mining algorithms.

This paper proposes a framework for interactive query formulation using metadata-related conditions; however, it does not consider the problem of actually identifying

constraints by defining new mining algorithms. The latter problem has been addressed in the literature under the form of mining sequential patterns [22–24] and association rules [25] or combinatorial pattern discovery [13]. If we consider paths of a semistructured database as sequences of labels, then we may apply well-known algorithms in order to find frequent subsequences of labels as well as (sequence or sibling) association rules. This is a vertical way of deriving sequences from a semistructured database, as opposed to the horizontal view of the database corresponding to DTDs. Mining association rules or sequential patterns for our purposes may either take place initially, or at the time of a request. In the latter case, our framework could benefit from algorithms in which users play an active role in guiding the mining process [26, 27].

Our framework is equally relevant and could benefit from previous work on discovering frequent tree expressions in documents [28, 29]. Typical structures in [28] can be used as a 'table-of-content' for gaining the general information of a source, or as a 'road map' for browsing and querying a source. The focus of our work is again not on mining such typical structures, but on demonstrating the use of several kinds of metadata information as guides in the querying process.

Finally, helping users find their way in a semistructured database has also been considered in [30, 31] but following a different approach. In particular, they have not considered structural properties and constraints as a means of adding flexibility to the querying interface, but instead exploited path information generated and stored in dynamic schemata (DataGuides). DataGuides are similar to DTDs in that they provide concise and convenient summaries of semistructured databases. Unlike static schemata, DataGuides conform to the data, rather than forcing data to conform to DataGuides. Hence, they always represent the current state of the database and play a significant role especially in contexts where it is implausible to specify and maintain a schema. DataGuide summaries may be used in our framework as concise and rich sources for mining metadata constraints efficiently.

7 Conclusion

In this paper, we proposed an interactive and adaptive model for querying semistructured databases. Starting from an IR-style query, the user may discover interesting schema information, and gradually refine his/her search to a conventional *select* query. Our model has a dynamic notion of context, starting from high level information to raw data and limiting the scope from the whole database, to local subpaths or subtrees. Adhering to the human sense of searching things, it is not only based on value or location criteria, but it equally exploits a variety of high level schema-related knowledge. This information is stored in a warehouse, which is maintained based on an event-driven mechanism. The proposed model should be very useful for searching semistructured databases, when results need to be filtered based on detailed criteria and, hence, low-granularity results (e.g. documents) are not acceptable. In general, it should be useful for all applications with an underlying semistructured database, but its benefits would be particularly obvious in the context of searches over the Web.

References

1. Cattell et al, R.: The Object Data Standard: ODMG 3.0. Morgan Kaufmann (2000)
2. Chaudhuri, S., Shim, K.: Optimizing queries with aggregate views. In: Extending Database Technology. (1996) 167–182
3. Chakravarthy, U., Grant, J., Minker, J.: Foundations of semantic query optimization for deductive databases. In: Foundations of Deductive Databases and Logic Programming. (1988) 243–273
4. Abiteboul, S.: Querying semi-structured data. In: Intl Conf on Database Theory. (1997) 1–18
5. Buneman, P.: Semistructured data: a tutorial. In: Symposium on Principles of Database Systems. (1997)
6. Evangelista-Filha, I., Laender, A., Silva, A.: Querying semistructured data by example: The QSByE interface. In: Intl Workshop on Information Integration on the Web (WIIW). (2001) 156–163
7. Buneman, P., Davidson, S., Fernandez, M., Suciu, D.: Adding structure to unstructured data. In: Intl Conf on Database Theory. (1997) 336–350
8. Nestorov, S., Abiteboul, S., Motwani, R.: Inferring structure in semistructured data. In: Workshop on Management of Semistructured Data. (1997)
9. Nestorov, S., Abiteboul, S., Motwani, R.: Extracting schema from semistructured data. In: ACM Intl Conf on Management of Data. (1998) 295–306
10. Kobayashi, M., Takeda, K.: Information retrieval on the web. ACM Computing Surveys **32** (2000) 144–173
11. Buneman, P., Fan, W., Simeon, J., Weinstein, S.: Constraints for semistructured data and xml. SIGMOD Record **30** (2001)
12. Buneman, P., Fan, W., Weinstein, S.: Query optimization for semistructured data using path constraints in a deterministic data model. In: Workshop on Database Programming Languages. (1999) 208–223
13. Wang, J., Chirn, G.W., Marr, T., Shapiro, B., Shasha, D., Zhang, K.: Combinatorial pattern discovery for scientific data: Some preliminary results. In: ACM SIGMOD Intl Conf on Management of Data. (1994) 115–125
14. Chandrasekaran, B., Josephson, J., Benjamins, V.: What are ontologies, and why do we need them? IEEE Intelligent Systems **14** (1999) 20–26
15. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems **16** (2001) 72–79
16. Zhong, N.: Ontologies in web intelligence. In: Practical Applications of Intelligent Agents, Springer. (2001)
17. Vianu, V.: A web odyssey: From codd to XML. In: ACM PODS Symposium on Principles of Database Systems. (2001) 1–15
18. Garofalakis, M., Rastogi, R., Seshadri, S., Shim, K.: Data mining and the web: Past, present and future. In: ACM CIKM'99 2nd Workshop on Web Information and Data Management (WIDM'99). (1999) 43–47
19. Chinenyanga, T., Kushmerick, N.: Expressive retrieval from XML documents. In: ACM SIGIR Conf on Research and Development in Information Retrieval. (2001) 163–171
20. Fan, W., Libkin, L.: On XML integrity constraints in the presence of DTDs. In: ACM PODS Symposium on Principles of Database Systems. (2001) 114–125
21. Fan, W., Simeon, J.: Integrity constraints for XML. In: ACM PODS Symposium on Principles of Database Systems. (2000) 23–34
22. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Intl Conf on Data Engineering (ICDE). (1995) 3–14

23. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: 5th Intl Conf on Extending Database Technology (EDBT). (1996) 3–17
24. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: PrefixSpan mining sequential patterns efficiently by prefix projected pattern growth. In: Intl Conf on Data Engineering (ICDE). (2001) 215–224
25. Srikant, R., Agrawal, R.: Mining generalized association rules. In: Intl Conf on Very Large Databases (VLDB). (1995) 407–419
26. Garofalakis, M., Rastogi, R., Shim, K.: SPIRIT: Sequential pattern mining with regular expression constraints. In: Intl Conf on Very Large Databases (VLDB). (1999) 223–234
27. Ng, R., Lakshmanan, L., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained association rules. In: ACM SIGMOD Intl Conf on Management of Data. (1998) 13–24
28. Wang, K., Liu, H.: Discovering typical structures of documents: A road map approach. In: ACM SIGIR Intl Conf on Research and Development in Information Retrieval. (1998) 146–154
29. Wang, K., Liu, H.: Discovering structural association of semistructured data. Knowledge and Data Engineering **12** (2000) 353–371
30. Goldman, R., Widom, J.: DataGuides: Enabling query formulation and optimization in semistructured databases. In: Twenty-third Intl Conf on Very Large Data Bases. (1997) 436–445
31. Goldman, R., Widom, J.: Interactive query and search in semistructured databases. In: First Intl Workshop on the Web and Databases (WebDB). (1998) 52–62