Integrating and Querying Web Databases and Documents

Carlos Garcia-Alvarado
University of Houston
Houston, TX 77204, USA

Carlos Ordonez
University of Houston
Houston, TX 77204, USA

ABSTRACT

There exist many interrelated information sources on the Internet that can be categorized into structured (database) and semistructured (documents). A key challenge is to integrate, query and analyze such heterogeneous collections of information. In this paper, we defend the idea of building web metadata repositories using relational databases as the main source and central data management technology of structured data, enriched by the semistructured data surrounding it. Our proposal rests on the assumption that heterogeneous relational databases can be integrated (i.e. entity resolution is assumed to work well) and thus can serve as references for external data. That is, we tackle the problem of integrating information in the deep web, departing from databases. We discuss a prototype system that can integrate and query metadata and related documents, based on relational database technology. Metadata includes database ER model elements like database name, table, and column (entity, attribute). Web document data include files, documents and web pages. Links between metadata and external documents are built with SQL queries. Once databases and documents are linked, they are managed and queried with SQL. We discuss an interesting scientific application of our solution with a water pollution database.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval—*Information Search and Retrieval*

General Terms

Algorithms, Experimentation

Keywords

Database, Documents, Integration

1. INTRODUCTION

Information retrieval (IR) and databases (DB) are two different major technologies for managing and searching for information that are slowly approaching each other. As stated in [3], almost any application that manages information requires the joint capabilities of IR and database systems to provide the users with a complete application [9]. The defining feature of a database is structure, and such structure has meaning (i.e. semantics), typically captured in an ER model. In contrast, in IR systems, there is some structure, but extracting the semantic information from the semistructured is a difficult task. Semistructured data sources (i.e. documents, web pages, files) are valuable for describing and enriching structured data. Managing and querying semistructured data sources is more challenging than self-described structured data (i.e. tables in a DBMS). Therefore, linking IR and DBMS is even more challenging.

We defend the idea of using a relational database as a central repository to store metadata and to integrate both technologies. The main advantage is that the integration can be done efficiently inside the DBMS by using SQL and extensibility features. The resulting repository can then be managed efficiently inside the DBMS due to the fact that the links can be represented in a relational form. With such motivation in mind, we built a novel hybrid DB-IR model integrating two previous separate approaches [8, 4]. Our new system finds and explores links between databases and documents, and then ranks them in a search engine fashion with a DBMS with primitive IR capabilities. Our system incorporates new optimizations, including efficient and approximate matching, link ranking and adapting the Rank-Join approach for top-k querying [6]. Our system enables discovering unknown links that go beyond simple keyword matching. Our goal is to find a core set of links between database metadata, database content, and documents, which can be efficiently managed and queried in a DBMS, instead of searching the document collection while lacking information links to the database. In addition, it is possible to extract information that it is not possible to find if both sources are queried independently.

We tested our application with a scientific database containing water pollution data and a collection of documents from the Texas Commission on Environmental Quality. The user can get insightful answers to non-trivial questions such as, "Which database tables are highly related to arsenic measurement?", "Which information in the database is related to a specific section of a document?" or "Which researchers query a particular table?"

2. LINKING AND QUERYING WEB DATABASES AND DOCUMENTS

Our approach can be divided into building the metadata repository (finding links), and ranking and querying this repository with relational queries.

http://doi.acm.org/10.1145/

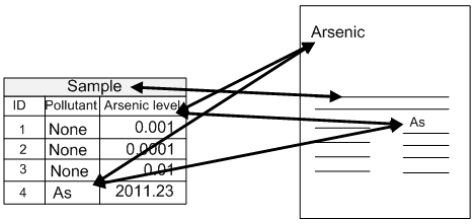

Figure 1: Links between database and documents.

L		E	
kid	k	kid	Database Location
1	sample	1	$\{T = sample\}$
2	arsenic	2	$\{T = arsenic\}$
...		2	$\{T = sample, c = pollutant, PK = 4\}$
5	pollutant	5	$\{T = sample, c = pollutant\}$
6	arsenic level	6	$\{T = sample, c = arsenic\ level\}$
...		...	

Table 1: Example of E and L.

2.1 Finding Links

Structured metadata is represented by table and column names. Database records (or rows) represent the basic atoms of information content, following a fixed table schema. Semistructured data are given by all the words contained in any external document, outside of the database. All documents are considered to have some structure that helps focus on a set of keywords in a particular portion of the document. This is particularly evident in web pages due to the fact that titles and paragraphs are specified by HTML tags. Additionally, documents' metadata, such as author, filename and title, can be extracted to establish additional links with this information. All of this information is then stored in the database and ranked based on discovered links from the matching process with a classical IR model. In summary, metadata and content from structured data and semistructured data are matched. Notice that assuming a perfect match scenario is unrealistic. As such, we decided to look for approximate matches. Three complementary types of links are generated: high, specific, and content-level links. These links differ in the level at which they match information metadata and data elements (see Figure 1). High level metadata links are based on describing a link between a table and a document. These links are found by finding matches between annotations describing the tables, as well as tablenames and information surrounding a document, such as filename, author, or even a description name, if available. Specific links relate sections or subsections in the document and columns in a table schema. Examples of these types of links are the keywords inside the header, <h2></h2> tags in HTML, and the column names. Finally, content links represent the most complex type of atomic granularity; they are the result of matching the content of structured data records and document keywords (e.g. keywords inside the tags).

In more detail, this matching is performed by exploiting a set of known concepts given by the user (e.g. arsenic, water pollution) and synonyms (e.g. {arsenic, AS},{pollutant, contaminant}), and then applying an efficient keyword matching algorithm in order to generate these links between parts of the database and certain parts of a document. As stated previously, we focus on approximate matching (e.g. keywords that match at least 90% of the given keyword) using the Approximate Boyer-Moore algorithm in order to obtain similar keywords (e.g. pollutant and pollutants). The main advantage of finding such links is that it is possible to explore integrated information at different granularity levels within a database, following the information on web documents, using high level links. Also notice that the generation of these links (or source integration) can be performed on a particular granularity. High level metadata links are generated much faster than specific links and in a similar fashion. Specific links are obtained much faster than content links. Also notice that the resulting number of links grows as we drill down into the integration of both sources.

When desired, the user can explore in more detail all documents related to a particular matched table or drill down into such a table. Thus, the user can query and explore both documents and the databases that relate to them.

loc. in db	keyword in db.	concept	doc.	keyword in doc.	loc. in doc.
High level					
db1:arsenic	arsenic	arsenic	1	arsenic	filename
db1:As	As	arsenic	3	As	metadata
Specific level					
db1:chemical	arsenic	arsenic	1	arsenic	title
db1:chemical	arsenic	arsenic	3	As	paragraph
Content level					
db1:sample: record1	As	arsenic	1	arsenic	title
db1:sample: record1	As	arsenic	3	As	paragraph

Table 2: Examples of high level, specific level, and content level links.

The main algorithms for performing such an exploration are detailed in [5], in which the main idea is to keep a list with all the unique keywords to integrate in both sources (given as a set of valid concepts and synonyms). These concepts are then matched to elements in the database (storing their location) and then performing a similar search within the corpus. An important characteristic is that when a keyword is not found in either the database or the corpus, this concept is dropped from the search list to avoid looking into the rest of the sources. Upon completion of this process, the set of approximate matches of both sources is matched to obtain the links. In order to avoid repeating computations, it is preferable to search for all the links (high level, specific links or content links) at the same computation.

2.2 Processing with Relational Queries

The basic IR system in SQL is introduced in [4]. The application was developed using SQL statements and in-DBMS extensibility mechanisms, such as User-Defined Functions (UDFs) for stemming, SQL statements for cleaning, storing and obtaining the links, and UDFs for parsing and ranking [4]. Notice that unlike [4], our ranking is focused on the newly discovered links and not on keywords. Therefore, links are weighted considering whether they match metadata or actual data content, as well as how close they match it (due to the approximate matching result). If links connect metadata elements, then the links can be weighted computing a measure summarizing actual content (similar to applying the Vector Space Model). If links are connecting content, then they are weighted based on the number of links associated with a particular concept (using a summarization). For example, if a link matches a document and a table, these two elements are composed of a set of keywords that can be used to compute a cosine similarity. On the other hand, when links are connecting a record in the database (in the

content level) and a keyword in a document, they will have to be ranked based on the number of links associated with that particular record. Once an appropriate ranking function has been defined, the retrieval of top-k links can be done efficiently (e.g. using the Rank-Join algorithm).

Links between the database and documents are built as follows: an input table with basic keywords (representing concepts), is given to perform matching. An auxiliary table is used to represent concept synonyms. These keywords are then matched between every element in the database and then for every keyword in the corpus. This matching is performed using user-defined functions in order to keep in-main memory all the data structures used to obtain the edit distance for every concept and every given string coming from the database or a document in the corpus. This matching step is the bottleneck of the link matching. However, to ease the processing, we perform the matching in batches. In other words, a set of elements in the DBMS or documents is scanned simultaneously to avoid I/Os. The size of the batches, as well as a valid approximate match, are parameters that need to be tuned by the user. A sample list of the concepts L and the elements found in the database E are presented in Table 1. Summary tables are created to avoid repeated searches on a particular data source and they include the location of the concept in the database and in the corpus. By using these summary tables, it is possible to build metadata and content links by performing a join on the matching keywords. The join on the matching keyword is performed by using a hash join on the concept id that was assigned during the matching phase. The resulting hash join is quite efficient due to the keyword indexes.

Notice that the type of link is based on the location of the elements in the database and the keywords in the corpus. By joining these two summary tables containing the concept match and the location of the match between the two sources, we can find that there are indeed two metadata links pointing to a table name and a column name. Also, during this phase, the concepts that do not have a corresponding link between the database and the documents must be removed from the list. Additionally, the number of links associated with particular concept must be computed (used to rank content links). A final step for link creation is assigning a weight to every content keyword that can be associated with a corresponding match in the corpus. By doing so, it is feasible to rank all metadata links.

Exploring (ad-hoc querying) links is less computationally expensive due to the links that have already been found, and thus work on materialized views. The querying is performed on these smaller indexed tables by using selection predicates p to find matching keywords given in a query, and the ranking is obtained by computing the corresponding weights for every concept (see Figure 2). Notice that p is a subquery that selects the keywords from the list L that approximately match a given concept. We should point out that ranking can be performed with any information retrieval ranking method. Due to their solid theoretical foundation, we used the vector space model and the Rank-Join algorithm for returning the top-k concepts. The Rank-Join algorithm was pipelined as an operation using a user-defined function and managed as an approximation for efficiency purposes.

3. APPLICATION ON SCIENTIFIC DATA

Our system is a Web DBMS Client Application with a client-server architecture. The processing works entirely inside the DBMS exploiting efficient SQL queries. These SQL queries are also enhanced by efficient extensibility features for complex processing (e.g. matching or parsing). Therefore, the front-end is a thin layer of SQL generation, which connects to the DBMS via ODBC and displays the final results. The front-end was developed with ASP .NET. The user interface allows searching documents utilizing the links involving Users, Tables, Documents, Columns and Fields, as well as the document abstracts. The output shows the results that are part of the database schema (e.g. columns), the results that are part of the semistructured world (e.g. authors) and the SQL associated to obtain those results. The front-end also allows the user to define the settings of the application, such as the document loading options, on-demand links search, and the top-k implementation (exact built-in top-k or UDF Rank-Join).

```
SELECT P.kid, R.i, E.level, P.d
FROM
   L_predicate P
   INNER JOIN database_keyword E
     ON  (P.kid = E.kid)
   INNER JOIN document_keyword R
     ON  (R.kid = E.kid)
ORDER BY  E.level, R.i, P.d DESC
```

Figure 2: Querying the link repository.

We are currently using our system on a scientific database containing water pollution data from wells in Texas (TWWD). The goal is to determine if water is polluted by humans or if it is naturally polluted. This database consists of 32 tables with up to 214 columns, where each table has an average size of 111,488 rows. Tables contain measurement information regarding chemicals found in water samples, such as concentration and units of measurement, as well as the geographical location of the well and associated descriptions and measurements of the results of several water samples. Due to the complexity of establishing semantics of numeric data, we are currently ignoring numbers during the linking process, but that is an area for future research. Associated with this database, we analyzed a pre-processed (stop words removed, stemmed) collection of documents describing scientific findings regarding the causes for water pollution.

In these two data sources, we applied two phases: (1) finding links and (2) navigating interrelated documents. Phase 1 is the most difficult and time consuming, especially while factoring in semantics. Phase 2 includes top-k selection, exploration of the links, and monitoring performance. An example of discovered links associated with the concept "arsenic" is shown in Table 2. In addition, in Table 3 we also show the resulting links found using the query presented in Figure 2. Our system allows more complex queries by weighting using the content of the metadata or faster alternatives such as the Rank-Join.

4. RELATED WORK

Heterogeneous source integration has been approached in similar research. EROCS is one of the few attempts at obtaining a linkage between structured and unstructured data [2]. EROCS focuses on linking transactions and then per-

kid	i	f	δ	Database Location
2	1	1	0	$\{T = arsenic\}$
2	2	1	0	$\{T = arsenic, c = pollutant, PK = 4\}$

Table 3: Ranking Links (query = "arsenic").

forming data mining operations over the newly discovered links. The linkage is performed using templates for identifying entities. However, unlike our approach, EROCS does not extend the DBMS functionalities using user-defined functions, nor does it deal with approximate matching. In [10, 7] the authors have a similar objective of linking structured and unstructured sources. This is probably the closest work to our research. However, in their approach, the DBMS is analyzed as a graph and then clustered. The final result is a traditional rank on top of the nodes of a graph. Similar to EROCS, their system does not take advantage of the extensibility features of the DBMS and avoids approximate matching. In addition, the ranking is performed on the extracted concepts and the weights are not assigned based on the frequency of appearance or location of these concepts.

Finally, in [8, 6], the authors suggest a method of relating schema-keywords with keywords in the documents, and generating macro and micro relationships. The matching between both sources is done using an Object Data Model, as presented in [1], and the unstructured sources were managed using XML. We integrated the schema matching and keyword search ideas to present a more complete solution on the existing types of links between elements and documents and present a novel idea on how to rank documents given these relationships within the DBMS. In addition, we do not require a given Object Data Model for discovering these links between the structured and unstructured sources, and our linkage is performed only with concept and synonym lists. Finally, this paper extends the ideas presented in [5] by limiting the high level links to relate only metadata, the specific links to relate certain sections of the keywords in the documents, and provides a ranking based on the content of the high or specific links. The objective of these modifications was to obtain efficient metadata linkage and allow a drill down exploration through more granular links.

5. CONCLUSIONS

In this paper, we presented a process for discovering "links" between the elements of a central database and a collection of documents. To the best of our knowledge, this research is one of the few efforts to obtain, manage, and rank links between these sources. The links are defined in different levels of granularity based on the elements they link. High links relate a document and a table, and specific links relate a column of a table and a section or subsection of a document. The content links match the actual rows of the database to specific keywords within the content of the semistructured source. In addition, an algorithm for discovering such relations was presented. We proposed an approximate search which provides a fair policy for obtaining matching concepts. We also found that that the bottleneck of the algorithm is in searching for the elements in the database when a set of keywords is given. In addition, the search of approximate keywords appears to be a target for future optimizations. The algorithms perform efficiently because the search is performed in small and indexed data structures containing the set of valid relationships and summarized tables. We also propose a data layout, indexing and querying strategies that optimized retrieval times.

Future work in this area includes thorough experiments for verifying the reliability of the extracted links. In addition to some comparisons with similar integration approaches and exploring a parallel implementation of the proposed algorithms, which can speed up the execution of both tasks (preprocessing and retrieval). The optimization for approximate string matching with multiple keywords appears as an interesting area of research. Finally, the combination of more complex weighting techniques of the elements in the database, as well as formalizing the complexity and computability of finding links, remain as open problems for future research.

Acknowledgments

This work was supported by NSF grants CCF 0937562 and IIS 0914861.

6. REFERENCES

[1] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *ACM Sigmod Record*, 28(1):54–59, 1999.

[2] M. Bhide, V. Chakravarthy, A. Gupta, H. Gupta, M. Mohania, K. Puniyani, P. Roy, S. Roy, and V. Sengar. Enhanced Business Intelligence using EROCS. In *Proc. of ICDE*, pages 1616–1619. IEEE Computer Society Washington, DC, USA, 2008.

[3] S. Chaudhuri, R. Ramakrishnan, and G. Weikum. Integrating DB and IR technologies: What is the sound of one hand clapping. In *Proc. of CIDR*, 2005.

[4] C. Garcia-Alvarado and C. Ordonez. Information retrieval from digital libraries in SQL. In *WIDM 2008*. ACM New York, NY, USA, 2008.

[5] C. Garcia-Alvarado and C. Ordonez. Keyword Search Across Databases and Documents. In *Proc. ACM SIGMOD Workshop on Keyword Search on Structured Data (KEYS)*, 2010.

[6] C. Garcia-Alvarado, C. Ordonez, and Z. Chen. DBDOC: Querying and Browsing Databases and Interrelated Documents. In *Proc. ACM SIGMOD Workshop on Keyword Search on Structured Data (KEYS)*, 2009.

[7] G. Li, B.C. Ooi, J. Feng, J. Wang, and L. Zhou. EASE: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In *Proc. of ACM SIGMOD*, pages 903–914. ACM, 2008.

[8] C. Ordonez, Z. Chen, and J. García-García. Metadata management for federated databases. In *ACM CIMS Workshop*, pages 31–38, 2007.

[9] L. Qin, J.X. Yu, and L. Chang. Keyword search in databases: the power of RDBMS. In *Proc. of SIGMOD*, pages 681–694, 2009.

[10] Q.H. Vu, B.C. Ooi, D. Papadias, and A.K.H. Tung. A graph method for keyword-based selection of the top-K databases. In *SIGMOD*, pages 915–926. ACM, 2008.