

# SearchWebDB: Searching the Billion Triples!

Haofen Wang<sup>1</sup>, Thanh Tran<sup>2</sup>, Peter Haase<sup>2</sup>, Thomas Penin<sup>1</sup>, Qiaoling Liu<sup>1</sup>,  
Linyun Fu<sup>1</sup>, and Yong Yu<sup>1</sup>

<sup>1</sup> Department of Computer Science & Engineering  
Shanghai Jiao Tong University, Shanghai, 200240, China  
{whfcarter, tpenin, lql, fulinyun, yyu}@apex.sjtu.edu.cn

<sup>2</sup> Institute AIFB, Universität Karlsruhe, Germany  
{dtr, pha}@aifb.uni-karlsruhe.de

**Abstract.** In recent years, the amount of structured data in form of triples available on the Web is increasing rapidly and has reached more than one billion. In this paper, we propose an infrastructure for searching the billion triples – called SearchWebDB – that integrates data sources publicly available on the web in a way such that users can ask queries against the billion triples through a single interface. Approximate mappings between schemata as well as data elements are computed and stored in several indices. These indices are exploited by a query engine to perform query routing and result combination in an efficient way. As opposed to a standard distributed query engine requiring the use of formal languages, users can ask queries in terms of keywords through SearchWebDB. These keywords are translated to possible interpretations presented as structured queries. Thus, complex information need can be addressed without imposing too much of a burden to the casual users.

## 1 Introduction

In recent years, the amount of structured data available on the Web has been increasing rapidly. Currently, there are billions of triples stored in web data sources of different domains. Likewise, these data sources become more tightly interrelated as the number of links in form of mappings is also growing. This development of a “data web” opens a new way for addressing complex information needs using web resources. Besides searching for documents that match keywords, the user can more specifically ask for pieces of information that satisfy some criteria according to structured query languages. However, the scale and the diversity of this data web put forward many challenges.

**Usability:** Searching the data web effectively requires the use of formal query languages. However, the burden of translating an information need into a structured query is a difficult task that should not be imposed on the casual users, as it would hinder the widespread exploitation of the data web. Instead, simple search paradigms adequate for the lay user are needed.

**Heterogeneity:** In order to fully exploit the data web, available data sources need to be managed in an integrated way. However, data sources cover different,

possibly overlapping domains. We encounter discrepancies on the schema-level, i.e. the way the data is modeled structurally, as well as the data-level, i.e. the way data values are represented.

**Imperfection:** Data contained in different sources might be redundant, complementary or conflicting. Definitely, recent advancements in entity consolidation as well as conflict resolution help address these different types of imperfections. While data integrity and quality can be greatly improved by applying these techniques, imperfection nevertheless can be seen as an inherent property of the data web. In terms of data web search, this means that there is a need for managing uncertainties.

With respect to these challenges, we propose an infrastructure called SearchWebDB that exhibits three main distinctive features:

**Expressive keyword search:** As opposed to the standard query engines (e.g. Sesame [1] and Virtuoso [4]) that require the use of formal languages, users can formulate queries in terms of keywords. Thus, no burden is imposed on the casual users. Using the same paradigm that has been proven to be useful for document retrieval on the web, the user can search and explore the data web through keywords. These keywords are translated to possible interpretations in terms of structured queries. Subsequently, the user selects the query that best fits his need. The selected structured query is then posed against the query engine. Note that this approach is different from systems (e.g. Sindice<sup>3</sup>, Watson<sup>4</sup>) that simply match keywords against an index of data elements. The results obtained using SearchWebDB do not only match the keywords but also satisfy structural conditions. In other words, SearchWebDB features a more expressive query capability.

**Integrated management of web data sources:** SearchWebDB integrates data sources publicly available on the web in a way such that users can ask queries against the data web in a transparent way. Existing techniques are exploited to compute mappings between schemata as well as data elements in an automatic way. Together with mappings already contained in the data sources, they form a data web containing of data sources that are linked at both the schema and the data level. While the schema level mappings are exploited by a query engine for distributed query processing (query decomposition and routing), the data level mappings are used for result combination.

**Handling of uncertainty:** Uncertainties are involved at many levels. For instance, expressing the information need in terms of keywords is imprecise. The quality and suitability of a data source might vary with respect to a given query. While data redundancy is partially resolved by the automatic computation of mappings, results cannot be assumed to be perfect. These different types of uncertainties are managed through an integrated ranking mechanism such that the returned answers represent the best approximation, given the imperfections in the data and the imprecision of the user keyword query.

---

<sup>3</sup> <http://www.sindice.com/>

<sup>4</sup> <http://watson.kmi.open.ac.uk/WatsonWUI/>

In Section 2, we present a motivating scenario along with its realization in SearchWebDB. We then discuss the underlying architecture of SearchWebDB in Section 3. Finally, we conclude the paper with discussions of future work in Section 4.

## 2 Search the Billion Triples with SearchWebDB

We illustrate the capabilities of SearchWebDB through the following example.

**Data web search scenario:** A person who is interested to learn about the Semantic Web is planning to attend the ISWC 2008 conference. She wants to prepare her visit by collecting information about the community, the people behind ISWC, including e.g. their research topics, relevant publications etc. While her information need may still be vague and she is far from able to express her needs in terms of a structured query, she might condense her information need into the simple keywords: “*ISWC topics people*”.

SearchWebDB integrates publicly available data sources such as WordNet, DBpedia, Yago, Freebase, Geonames, US Census, SwetoDBLP<sup>5</sup>, semanticweb.org<sup>6</sup> and SWRC<sup>7</sup>. Among them, the relevant data sources that may contribute to potential answers are SWRC, semanticweb.org and SwetoDBLP. Semanticweb.org contains information about the ISWC conference, SWRC comprises of data about the research group that organizes the ISWC 2008 along with FOAF profiles of its affiliates and the SwetoDBLP dataset provides bibliographic metadata about papers published at the ISWC conferences. Given these data sources, there are various interpretations of the keywords, that might provide useful answers to fulfill the user’s information need, e.g.:

- What are the research topics of the people organizing ISWC?
- What are the topics and authors of the papers presented at ISWC?

These interpretations correspond to the following structured queries:

```
SELECT ?topic ?person
WHERE http://semanticweb.org/id/ISWC2008 Has_OC_member ?person
      ?person rdf:type Person
      ?person worksOnTopic ?topic
```

```
SELECT ?topic ?person
WHERE ?publication dc:creator ?person
      ?person rdf:type Person
      ?publication venue http://semanticweb.org/id/ISWC2008
      ?publication isAbout ?topic
```

---

<sup>5</sup> <http://www4.wiwiw.fu-berlin.de/dblp/>

<sup>6</sup> <http://semanticweb.org/wiki/ISWC2008>

<sup>7</sup> <http://www.aifb.uni-karlsruhe.de>

In this scenario, we observe that interpretations might result in queries that can only be answered by combining results from different data sources. For example, the fact that Rudi Studer is the local chair of the ISWC 2008 conference is stated at semanticweb.org, while his research interests are stated in his FOAF profile (contained in SWRC). Similarly, SwetoDBLP does not explicitly classify topics, while this classification would be available from other sources.

We will now describe how the scenario is realized in the SearchWebDB system from a user perspective.

**Interaction with SearchWebDB:** The user interacts with SearchWebDB via a web interface available at <http://searchwebdb.apexlab.org>. The interface has two views: one for Query Translation and the other is for Search Refinement.

The Query Translation view features a keyword query interface, which can assist the user in typing keywords. This is extended to “phrase completion” so that when the first keyword has been entered, a list of phrases are presented to the user for selection. After entering keywords, the system computes a list of candidate structured queries and presents them to the user.

When the user chooses a query, the Search Refinement view is presented. This view consists of three panels. The Result Presentation Panel shows a list of ranked results. The Refine Panel can be used to modify the selected query. Note that a query might involve different entity types (e.g. `person` and `publication`). Given the entity type selected by the user, this panel shows the facets, i.e. properties that can be used to refine entities of this type. The History Panel visualizes the different steps occurred during this process of iterative query construction using keywords and facets.

### 3 SearchWebDB Infrastructure

In this section, we describe the infrastructure underlying the SearchWebDB system. Fig. 1 shows a diagram of the conceptual architecture. It supports complex information needs expressed in keywords, using open web data sources that are integrated through approximate mapping definitions. In particular, RDF data sources are supported – which in the following, will be referred to as *data graphs*. In the architecture, we can distinguish between components supporting *offline* and *online* processes: *Graph Data Preprocessing* refers to the offline process where the data graphs are preprocessed and stored in specific data structures of the *Internal Indices* (keyword-, structure- and mapping index). These indices are the result of different preprocessing steps – relying on *Lexical Resources* – applied on the data graphs for efficient online query processing. *Keyword Translation* focuses on translating keywords to query graphs – intermediate representations of the user information need from which conjunctive queries will be derived. These queries might cover multiple data sources. They represent the different interpretations of the keywords, which are presented to the user. Finally, *Distributed Query Processing* returns the top-*k* ranked results w.r.t. the user selected query. This processing involves decomposing the query into subqueries that can be evaluated via *Local Query Processing* by the individual *Web Data Sources*.

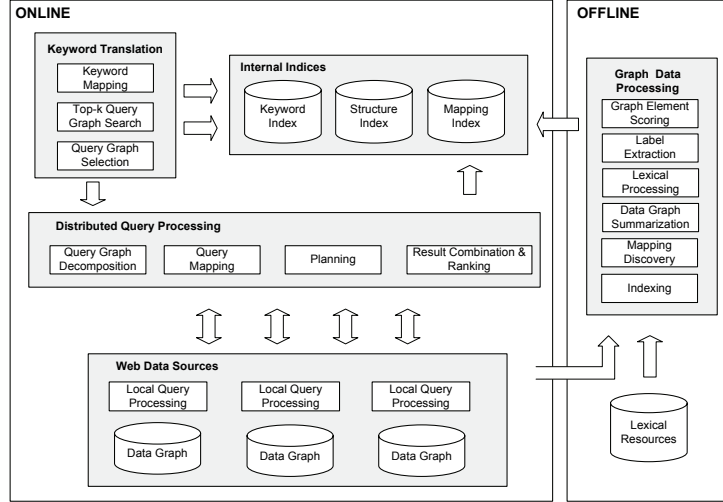


Fig. 1. SearchWebDB Infrastructure

### 3.1 Data Preprocessing

In order to perform online query processing in an efficient way, we perform a preprocessing of the data graphs to construct several internal indices:

**Keyword index** Firstly, the labels of elements of the data graphs are *extracted* and finally stored in the keyword index. Given a keyword query, this index can then be used to identify the elements in the data graph matching a keyword. Each matching element in the index is associated with additional information. The matching score is exploited in query ranking. Further, the keyword index contains data source information, which allows for routing partial queries to the local query engines that can answer them.

**Structure index** The structure index employed here is similar to the one proposed in our previous work. Namely, we have shown in [5] that keyword search is more efficient by at least one order of magnitude in most cases when performing graph search on a structural summary only, instead of using the entire data graph. These summaries, called schema graphs, are stored in a *structure index*, which is later used for searching query graphs. If a schema for a data graph is incomplete or not available in advance (as it is often the case with RDF data), it can be computed via summarization techniques from the data [5].

We extend the notion of schema graphs as structural summaries of single data graphs to *integrated schema graphs* for the multi data sources scenario. An integrated schema graph is a set of schema graphs plus correspondences between elements of these individual schema graphs (called schema level mappings).

**Mapping index** We employ tools to discover mappings in an automatic manner. These mappings are stored in the mappings index. In particular, we implement the process as discussed in [3]. It can be decomposed into 1) en-

gineering of similarity features, 2) selection of candidates, 3) computation of similarities 4) aggregation of similarities, and 5) derivation of correspondences based on the aggregated similarity values. During this iterative process, mappings are derived from syntactic, structural and semantic similarities between candidate elements. As a result, about 1,500 schema level mappings (mappings between classes and properties) and more than 5 million data level mappings (mappings between individuals) are discovered for the data sources currently available in the system (c.f. Section 2). We observe that DBpedia and Freebase are connected most tightly as more than 90 percent schema level mappings and half of the data level mappings are found between the two data sources.

In SearchWebDB, schema level mappings are used to connect schema graphs. They are exploited during query translation (for “routing” keywords to suitable data sources). Data level mappings are incorporated in the process of result combination, i.e. to perform joins.

*Discussion* In contrast to local- and global-centric approaches to data integration, there is no assumption of a mediated schema in our approach. Constructing and maintaining a mediated schema that provides a shared vocabulary for all resources on the highly dynamic Web environment is difficult [2]. Also, the complexity and overheads in mapping local schemas with the mediated schema are not affordable. In our approach, mappings might exist between any pair of data sources on the Web. Further, the notion of mapping employed in our approach is more general. It is not only restricted to schema elements but might also captures correspondences between data elements.

### 3.2 Keyword Translation

Using information in the keyword index, keywords are firstly *mapped* to a set of keyword elements. This is to find out whether the issued keywords can be answered using the available Web data. From these keyword elements, the information in the structure index are augmented and then explored to find the top- $k$  query graphs. Each query can be seen as an alternative interpretation of the user information need expressed in keywords. These alternative interpretations are sorted according to a *query ranking* scheme, and presented to the user for final *selection* and *refinement*. Unlike traditional keyword search which computes the top- $k$  answers, the data web search process supported here involves query presentation as an additional intermediate step. As discussed in [5], the computed structured queries can facilitate comprehension of the results and also, enable more precise refinement than using keyword queries. According to user feedbacks from an initial usability study, 90% (12 participants in all) prefer to obtain the queries first, rather than the answers directly. Also, all users prefer to do refinement on the structured queries, rather than on the keywords.

**Computation of top- $k$  query graphs** The top- $k$  procedure [5] for this starts from the keyword elements and iteratively traverses all distinct paths from these elements. During this procedure, the path with the highest score so far is selected for further exploration. At some point, an element might be

discovered to be a connecting element, i.e. there exist paths from that element to all keyword elements. These paths are merged to form the query graph. The process continues until the upper bound score for the query graphs yet to be explored is lower than the score of the  $k$ -ranked query graph already computed. While this procedure has been proposed for a single graph, it is straightforward to extend it to the data web scenario. In fact, we treat the mappings between data sources just like normal edges, i.e. they can be traversed during exploration to construct the *integrated query graph*. The special semantics of mappings and other uncertainty aspects resulting from the data web scenario are dealt with in the scoring process.

**Scoring query graphs** The computation of the top- $k$  query graphs relies on a scoring function. It incorporates three different aspects: 1) the importance of graph elements, 2) the matching score of keyword elements (captures imperfection in the keyword mapping), and 3) the length, where queries of shorter lengths are preferred due to the assumption that closely connected entities more likely match the information need. Further, schema graphs that cover a large number of keywords are prioritized in the search. This is in line with the intuition that a query should to be processed using as few data sources as possible.

### 3.3 Distributed Query Processing

Distributed query processing starts with the integrated query graph as selected by the user. The graph is *decomposed* into parts such that each can be processed against a particular data source. Before *routing*, each part (subgraph) is *mapped* to a conjunctive query and *translated* to the query language (e.g. SQL, SPARQL) supported by the local query engine. In our implementation, we use Semplore [7] as the triple store and query engine for local query processing in SearchWebDB. It enjoys the benefit of spatial locality for fast access and achieves good scalability due to its trade-off on query capability. For optimizing performance, a *planner* is employed to determine an appropriate order of query execution. Finally, the results obtained from the local data sources are *combined* to arrive at a list of top- $k$  results sorted according to their scores. In particular, a set of *union* and (or) *join* operations are performed on the tuple sets obtained from the local query engines.

**Query decomposition** As discussed previously, an integrated query graph contains two types of edges. There are edges belonging to a particular schema graph, and edges that connect elements of two schema graphs (i.e. schema level mappings). According to this structure, query decomposition can be accomplished by simply omitting all mapping edges from the query graph. The resulting query graph is a set of strongly connected components – each corresponds to a query part.

**Query mapping** Since every partial query graph contains only elements that belong to a particular schema graph, the mapping procedure for single data source keyword search as proposed in [5] can be used. Basically, edges are mapped to predicates whereas vertices are mapped to variables and terms of the conjunctive query. In fact, a query part is not mapped to a conjunctive query but

the syntactic variant supported by the local query engine. For running queries against a database engine, a query part is mapped to a SQL query. When the data source is a RDF store, it is translated to a SPARQL query.

**Result combination** A key operation in this context is, given two data sources, identify all pairs of entities (tuples) that are (approximately) the same. Here, information stored in the mapping index are leveraged to combine (join or merge) the results from local query processors. During this process, scores of data tuples are propagated and aggregated according to an *answer ranking* scheme proposed in [6].

## 4 Conclusions and Future Work

We provide an infrastructure SearchWebDB that enables integrated search across multiple sources in the data web. Currently, we are able to manage, i.e. store, retrieve and query data in the scale of billion triples from the data sources available in the system.

In building SearchWebDB, we have extended our existing work on large scale triple stores and keyword search. In the resulting system, we are able to deal with heterogeneity and imperfection in the data, while providing a user interface that follows the simple keyword paradigm and hides the underlying complexity from the user, yet still provides the expressive power of structured queries against the data web.

We have also identified several areas of future work. First, additional cleaning should be performed on the other data sets provided by those semantic Web search engines so that the quality of involved triples can be guaranteed. Second, incremental approximated mapping in a pay-as-you-go manner should be supported due to the rapid growth of semantic data. Finally, we expect more challenges to arise once we move from a fixed (albeit large) set of data sources – as in the Billion Triple Challenge – to an open and wild data web.

## References

1. J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *ISWC*, 2002.
2. N. Choi, I.-Y. Song, and H. Han. A survey on ontology mapping. *SIGMOD Record*, 35(3):34–41, 2006.
3. M. Ehrig and S. Staab. Qom - quick ontology mapping. In *International Semantic Web Conference*, pages 683–697, 2004.
4. O. Erling and I. Mikhailov. Rdf support in the virtuoso dbms. In *CSSW*, pages 59–68, 2007.
5. T. Tran, H. Wang, S. Rudolph, and P. Cimiano. Efficient computation of formal queries from keywords on graph-structured rdf data. In *To appear at ICDE*, 2009.
6. H. Wang, T. Tran, and C. Liu. Ce<sup>2</sup> – towards a large scale hybrid search engine with integrated ranking support. In *To appear at CIKM*, 2008.
7. L. Zhang, Q. Liu, J. Zhang, H. Wang, Y. Pan, and Y. Yu. Semplore: An ir approach to scalable hybrid query of semantic web data. In *ISWC/ASWC*, pages 652–665, 2007.