

Advances in Data Management - Web Data Management

A.Poulovassilis

1 Introduction

In this lecture I focus mainly on data management challenges and techniques relating to the integration and querying of Web data.

For a fuller list of topics that fall into the (very broad) area of “Web Data Management”, see the Topics of Interest of the workshops on “Web Information and Data Management” (WIDM) (ACM WIDM 2008 at <http://widm2008.comp.nus.edu.sg/>, ACM WIDM 2009 at <http://widm2009.ist.psu.edu/>), which include:

- Models and Languages for Web Data Management: Semi-structured Data, XML, RDF, Query Languages, Annotations, Ontologies
- Methodologies: Data Integration, Archiving, Security, Trust
- Tools, Infrastructure and Architectures: Web Site Design, Web Visualization Tools, Intelligent Agents on the Web, Web Services, Performance of Web Applications, System Design, Caching and Indexing of Web data, P2P systems
- Web Applications: Digital Libraries, Web Portals, Warehousing, Web Information Filtering, Web Commerce, Web Monitoring
- Web Mining: Web Usage Mining, Classification, Clustering, Resource Discovery, Personalisation, Web Data Extraction, Web Structure Mining

2 Preliminaries - Database support for Web applications

Databases sit at the back-end of many Web applications.

All the major DBMSs support one or more Application Programming Interfaces (APIs) which programs and 3rd-party tools can use to connect to databases and execute query/update statements on them e.g. ODBC, JDBC.

The flow of data between the Web Browser and the back-end database is typically as follows:

1. The HTML FORM construct is used to communicate data from the Web Browser to the Web Server, e.g.:

```
<FORM ACTION="newCustomerRegistration.jsp" method="post">
  <INPUT TYPE="text" NAME="FirstName"> <BR>
  <INPUT TYPE="text" NAME="LastName"> <BR>
  <INPUT TYPE="text" NAME="address"> <BR>
  <INPUT TYPE="submit" NAME='Submit' VALUE="Submit">
</FORM>
```

Any client-side processing logic required (e.g. for data validation) can be undertaken by using Javascript embedded within the HTML, or by using a Java applet.

Communication between the Web browser and Web server uses the `http` protocol.

2. The validated data from the form is passed to a program P on the web server (e.g. a Java servlet). This processes the data, possibly invoking additional programs on a separate application server to undertake further processing logic (e.g. using J2EE and EJBs).
3. These programs may generate queries or updates that are submitted to the back-end database for execution. The results from these requests will be passed back to program P .
4. P processes these results and dynamically generates an HTML results page which is sent back to the web browser for display.

(For details of the languages and protocols involved in supporting web applications, please see the Component Based Development and Internet & Web Technologies modules. See also Chapter 6 of Database Management Systems, R.Ramakrishnan and Johannes Gehrke, 2003.)

3 Web data integration

Traditionally, the web has provided vast amounts of information in *unstructured* form (i.e. text). Searching for this information has utilised techniques from Information Retrieval, in conventional search engines such as Google.

In recent years there has been an increase in the amount of *structured data* available on the web, arising from data that is stored in databases and is accessed via HTML forms — this is known as the *deep web*.

This data presents the same kinds of problems as we have discussed earlier in the context of heterogeneous database integration *but on a much larger scale* e.g. the back-end databases use different modelling languages, different data types, and different representations for the same real-world concepts.

These heterogeneities are manifested in the HTML forms provided for accessing the data, and in the results returned in response to user queries submitted via such forms.

This poses the question: is it possible to extend techniques that have been developed for heterogeneous database integration to access this kind of heterogeneous web content through a single query interface?

3.1 Domain-specific meta-search engines

One area of research in this direction is techniques for automatically generating **domain-specific meta-search engines**:

These aim to provide domain-specific meta-search utilising multiple source search engines.

One of the major steps involved in providing such functionality is the task of *schema matching* and *schema mapping*.

Note that in this context, the “schema” is the format of the HTML form that allows users to access content retrieved via each source search engine.

Schema matching identifies correspondences between elements from different schemas (these correspondences may be 1-1, 1-n, n-1, n-m).

Schema mapping defines these correspondences i.e. generates the view definitions that link two schemas.

The volumes and heterogeneity of Web data require the development of *automatic* schema matching and mapping techniques.

In the approach by Naz et al. 2009 (see the paper) a domain ontology serves as a “global schema”, allowing semantic heterogeneities between different search engine interfaces to be resolved by creating mappings between these interfaces and the domain ontology.

Several schema-matching techniques are used to automatically resolve semantic heterogeneities between the source search engine schemas: these are a variety of *element-level*, *structure-level* and *ontology-based* techniques (see the paper).

The mappings between the source search engine schemas and the domain ontology are also derived automatically.

These mappings are subsequently used to:

- generate an integrated search query interface;
- support query processing in the meta-search engine;
- resolve semantic conflicts arising during result extraction from the source search engines, so as to present an integrated, harmonised set of results to the user.

3.2 Web-scale data integration

Beyond domain-specific techniques, the PAYGO approach proposed by researchers at Google aims to address data integration on the scale of the whole web, not just for one specific domain (see paper by Madhavan et al.):

- In PAYGO, deep web content is discovered by simulating form submissions, retrieving results pages, and adding these results pages to the traditional web index (in an off-line fashion); this is called *surfacing* the web content
- with this approach, existing web content indexing technology can be reused; moreover searching is not dependent on the run-time characteristics of the underlying sources
- there is no single global schema encompassing all this deep web content of course, and instead it is envisaged that sets of source ‘schemas’ will be clustered into ‘topics’ over time; some of these schemas will serve as ‘hubs’, playing the role of integrated schemas
- the mappings between schemas will typically be approximate, not exact
- the entire integration approach is “**pay-as-you-go**”, in the sense that data sources will become increasingly integrated over time — this is in contrast to the “single-shot” integration in conventional data integration
- end-user queries are posed as keywords (not in a structured query language such as SQL or XQuery) and automatically routed to relevant sources
- query results are ranked, using their degree of match to the user’s query and prior knowledge about the user’s preferences

4 Linked data on the web

Large volumes of structured data that were hidden up to now in the deep web are beginning to be published as sets of RDF triples, in the form of *Linked Open Data* (LOD) — see <http://linkeddata.org/>

Ideally, common object identifiers in the form of URIs are used across linked data sources that are in the same application domain.

Different sets of RDF triples are published by different data providers, but referencing common URIs which provide links between these different data sets.

Federated querying services allow queries to be answered drawing results from multiple data sources.

For example, in the paper by Langegger et al., the SemWIK (Semantic Web Integrator and Query Engine) middleware aims to support data sharing in distributed scientific communities:

- An RDF Mapping Service – specifically, D2R Server (www4.wiwiss.fu-berlin.de/bizer/d2r-server/) – allows the content of each non-RDF data source to be queried as though it were RDF.
- The global ontology is expressed in OWL. GAV views map terms in this ontology to the data sources' RDF representations.
- Users' queries are expressed with respect to the ontology, using the SPARQL query language (www.w3.org/TR/rdf-sparql-query/).
- Using the GAV views, the SemWIK Query Processor reformulates user queries submitted to the ontology into sub-queries expressed on the data sources' RDF representations (query reformulation is by means of view unfolding; there is no support for reasoning on the ontology).
- These sub-queries are directed to the RDF Mapping Services over the data sources, which translate the SPARQL sub-queries into native queries that are submitted to the actual data sources for evaluation.

Homework

Read the paper by T.Naz et al. on “A Hybrid Approach to Schema and Data Integration for Meta-search Engines”. Technical Report BBKCS-09-02, Birkbeck, February 2009. Accessible at <http://www.dcs.bbk.ac.uk/research/techreps/2009/bbkcs-09-02.pdf>

Other references (for interest):

J.Madhavan et al. “Web-scale Data Integration: You can only afford to Pay As You Go”, Proc. CIDR 2007. Accessible at <http://www.cidrdb.org/cidr2007/papers/cidr07p40.pdf>

A semantic web middleware for virtual data integration on the Web, A.Langegger, Wolfram Woss and Martin Blochl, Proc. ESWC 2008, pp 493-507. Accessible at <http://www.langegger.at/papers/ESWC08-paper244.pdf>