Lightweight Databases

SA Dobson
Computing and Information Systems Department,
Rutherford Appleton Laboratory, UK
sd@inf.rl.ac.uk, http://www.cis.rl.ac.uk/people/sd/contact.html

VA Burrill
Computing and Information Systems Department,
Rutherford Appleton Laboratory, UK
vab@inf.rl.ac.uk, http://www.cis.rl.ac.uk/people/vab/contact.html

Abstract:

Current World Wide Web technologies concentrate on presenting documents to human readers. Although HTML identifies structures within a document, it does not allow the semantic content of document sections to be specified explicitly. We investigate a small extension to HTML which allows parts of a document to be mapped onto an underlying database schema. This allows automatic identification and extraction of key information from a web using standard database techniques. Such "lightweight" databases may span servers, with searches being performed at client- or server-side. We have applied this approach to generating "flattened" versions of hypertext documents suitable for printing.

Keywords:

Databases, resource discovery, semantic mark-up, HTML extensions, printing hypertext.

Introduction

The World Wide Web (WWW) allows hypertext documents to be created and read by users across the Internet. A huge amount of information is now available at the various WWW sites, and seems to be growing exponentially. Moreover the web is being cross-linked to enable users to find their way around the available information.

Although WWW documents may be useful to human readers, there is often a need to perform automated queries on information. Typical examples include searching for contact information, determining costs of services, re-formatting existing documents, searching indexes *et cetera*. The ability to perform predicate queries on a body of information is important for resource discovery and managing the information overload which WWW engenders. Current WWW technologies do not provide adequate support for such automated access to information.

Many authors have proposed the integration of relational database management systems (RDBMSs) with WWW, although most have concentrated on interactive rather than automated query submission. We have investigated an alternative approach in which web pages themselves are marked-up with semantic information according to an underlying database schema. We include the database structure explicitly into the web, allowing queries to be performed client- or server-side. Although not suitable for applications with highly intensive database requirements, such "lightweight" database webs allow modest database functionality to be obtained very easily and without the overhead of using a full RDBMS.

In this paper we present our approach and discuss it in relationship to other work on database integration and semantic mark-up. We describe a sample application of the technique, that of "flattening" a document with hyperlinks to generate a paper copy.

Semantic Mark-up

A document is a source of information for a reader. A typical document will be structured into headings, sections, tables *et cetera*, each of which may contain useful information. SGML[SoftQuad] allows a document's structure to be completely divorced from its presentation. HTML[Berners-Lee94] is less flexible in this respect, in that presentational information is implicit in the tags used to mark-up a document.

Although a document makes information available to the human reader, making the same information accessible to automatic processing tools requires additional effort. For example, although my telephone number is immediately available on the web, an automatic tool would be hard-put to distinguish it from my fax number. Human readers are far more tolerant than machines.

A tool must be able to determine the semantic content of part of a document and extract the relevant information (without extraneous formatting). It must be able to navigate around a web to locate the required information – preferably without lengthy searching – and deal with relationships between pieces of information.

Techniques already exist within the database world for formally describing interconnected information. The most widely-accepted technique is to use the relational model[Date86] to structure information. The information may then be accessed by posing queries in relational algebra.

We have investigated importing the relational formalism into WWW by means of a minimal extension to HTML. This allows elements of a document to be identified with elements of an underlying relational database. The information is still stored directly in the web, with the database structures offering an alternative view onto the data.

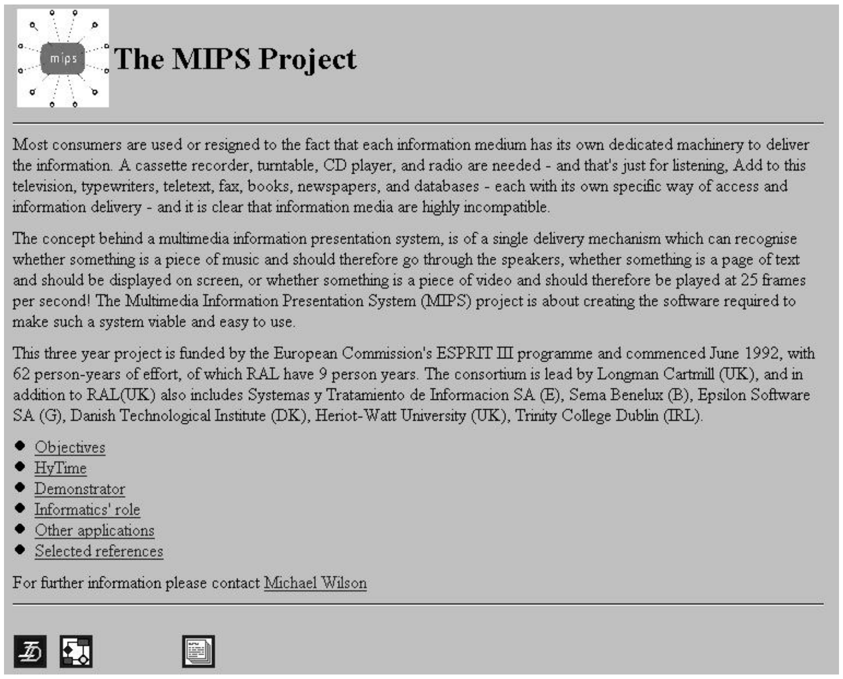
Notation

In overview, the approach is as follows: we first decide on the information which we wish to publish in machine-readable form, and perform a standard data analysis on it. The result of this is a conceptual model of the information as a collection of entities having attributes and connected by relations. We then mark-up information in the web pages according to this schema – note that it is not necessary to generate a logical model of the information as tables as would be the case for a standard relational database.

The notation for mark-up extends HTML with three extra elements:

- <ENT>...</ENT> identifies an entity of a particular class
- <ATTR>...</ATTR> identifies a named attribute of an entity
- <REL>...</REL> denotes a relationship between entities

These elements directly capture the conceptual-level structure of information, allowing individual entities to encapsulate several sub-values as attributes and allowing entities to be linked using meaningful connectives.



The MIPS Project

Most consumers are used or resigned to the fact that each information medium has its own dedicated machinery to deliver the information. A cassette recorder, turntable, CD player, and radio are needed - and that's just for listening, Add to this television, typewriters, teletext, fax, books, newspapers, and databases - each with its own specific way of access and information delivery - and it is clear that information media are highly incompatible.

The concept behind a multimedia information presentation system, is of a single delivery mechanism which can recognise whether something is a piece of music and should therefore go through the speakers, whether something is a page of text and should be displayed on screen, or whether something is a piece of video and should therefore be played at 25 frames per second! The Multimedia Information Presentation System (MIPS) project is about creating the software required to make such a system viable and easy to use.

This three year project is funded by the European Commission's ESPRIT III programme and commenced June 1992, with 62 person-years of effort, of which RAL have 9 person years. The consortium is lead by Longman Cartmill (UK), and in addition to RAL(UK) also includes Systemas y Tratamiento de Informacion SA (E), Sema Benelux (B), Epsilon Software SA (G), Danish Technological Institute (DK), Heriot-Watt University (UK), Trinity College Dublin (IRL).

- Objectives
- HyTime
- Demonstrator
- Informatics' role
- Other applications
- Selected references

For further information please contact Michael Wilson

Figure 1: A sample project page

None of the elements has any presentational content – they provide meta-information about part of a document but do not alter its presentation. They may thus be ignored by browsers presenting a document to a user. The overhead in adding such mark-up to a page is typically very small, and so does not impact on performance.

The elements turn a simple web into a *lightweight database*. They allow a database formalism to be imported directly into web pages, which in turn allows a database-aware client application to extract information directly from the pages according to the underlying schema.

We shall explain the use of the mark-up by means of an example below. Further details of the notation may be found in [Dobson94a].

Mark-up Example: Project Descriptions

To illustrate the use of the notation, we shall mark-up a page which describes a research project. Within our own Department all on-going projects have descriptions in the web. A typical project page is shown in figure 1. Each project includes a number of useful pieces of information, including:

- a long title, short title and logo;
- an introduction and the project's aims;
- any collaborating organisations;
- the funding source, total funding and RAL's share of it; and
- a contact name for further information.

We begin by defining a suitable conceptual model for a project. Since our projects are very similar, it is possible to derive a canonical model which encapsulates the core information: this still allows individual projects to include extra information on their pages if desired. The model is shown in figure 2 – note that each entity class has a *primary key* (indicated by underlining) which identifies each instance uniquely within a class.

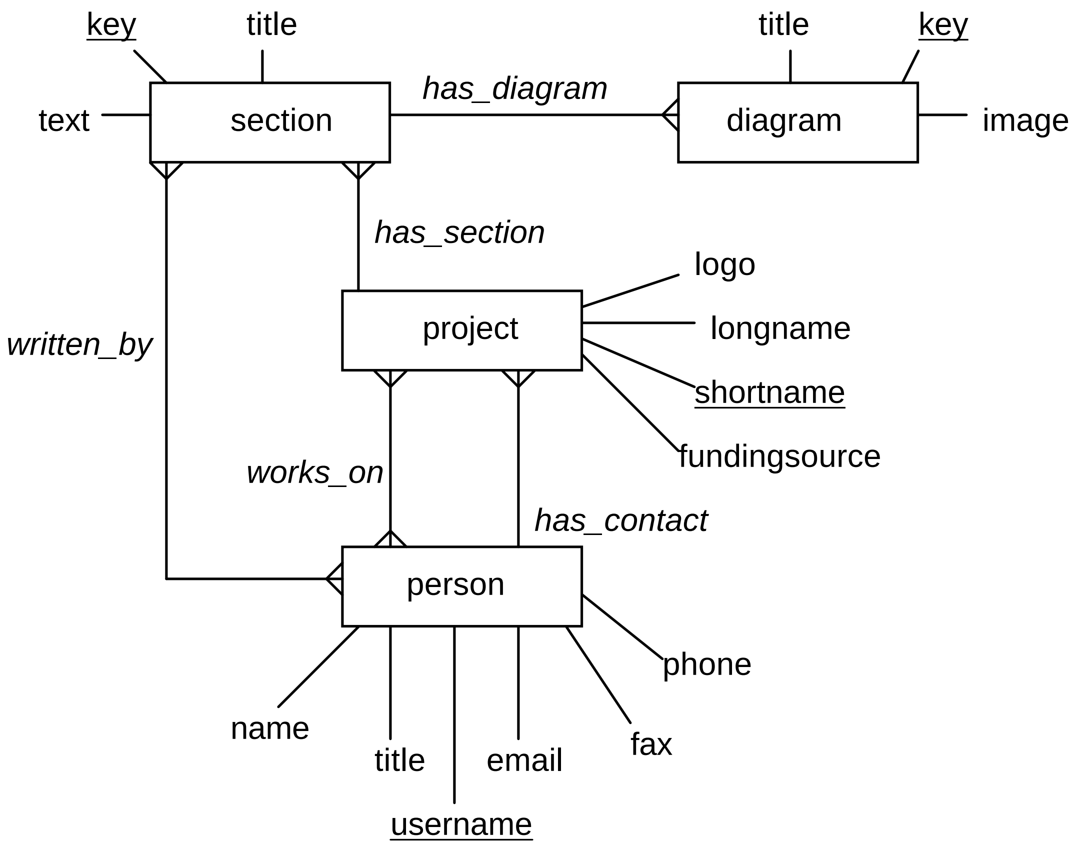


Figure : Conceptual model of project descriptions

Each project is represented as a single entity of the *project* class. This class has attributes of long and short names, logo *et cetera*. We mark-up the top-level project entity as follows:

```
<ENT KEY=mips CLASS=project>
<H1>
<ATTR NAME=logo><IMG SRC="MIPS.gif"></ATTR>
The
<ATTR NAME=shortname>MIPS</ATTR>
Project
</H1>
...
</ENT>
```

The ENT element provides the primary key value and entity class of the information. Significant pieces of information are captured within ATTR elements, which identify attributes by name. The page itself contains extra formatting information, such as the use of H1 for the heading, but this is not significant (from a database point of view) and lies outside the scope of the identified attributes.

Entities are linked using relations. In our example, the MIPS project is linked to a particular person under the *has_contact* relation. We may encode this as:

```
<ENT KEY=mips CLASS=project>
...
<REL NAME="has_contact" CLASS=person KEY=mdw
HREF="/people/mdw.html">
<A HREF="/people/mdw.html">Michael Wilson</A>
</REL>
...
</ENT>
```

The REL element relates the containing entity to another entity identified by key value and entity class under a named relationship. The HREF attribute provides location information, indicating the URL at which the target entity may be found. This simplifies searching, as a search engine may move directly to the target of the relationship. We regard this as a useful navigational hint, not as an essential part of the mark-up.

Notice that the REL element contains a denotation of the relationship, which in this case is a hyperlink. It is important to realise that the *relationship* structure of a lightweight database is distinct from the *hyperlink* structure of a document. One may form relationships between entities without using hyperlinks, or hyperlinks without invoking a relationship (in the database sense). This is especially useful when we want to provide information in the database which is not directly accessible by a human browser: a relationship without a corresponding hyperlink is effectively invisible, although a database-aware client could still traverse the relationship to acquire the information.

We may also wish to relate entities which are stored in the same document but which are logically distinct. For example, the MIPS project's introduction is a *section* entity presented along with the project's top page. This relationship may be encoded by nesting an entity element directly within a relationship element:

```
<ENT KEY=mips CLASS=project>
...
<REL NAME="has_introduction">
<ENT KEY="mips_introduction" CLASS=introduction>
Most consumers are...
</ENT>
</REL>
...
</ENT>
```

Again, the database structure of the information is separate from the way it is presented in terms of in-line inclusion and hyperlinks. One may see the information as being stored in a way which is most convenient for the most common mode of use (browsing) whilst remaining accessible to other modes (database queries).

Application Example: Printing Hypertext Documents

One of our motivations for the lightweight database extensions was a need to generate printed "hand-outs" from hypertext project descriptions for distribution to interested parties without web access. We have developed a small server-side application which uses the lightweight mark-up to "flatten" a document for printing.

The printing application is composed of three parts: an interpreter for the semantic mark-up, a query engine, and a template post-processor. The first stage interprets a lightweight database web to generate a table of entities, attributes and relationships. This table may then be interrogated by the query engine to extract the desired entities or attributes, using general database-style queries.

Templates are used to re-format extracted elements into HTML for presentation to the user. A template is an HTML document augmented with queries acting as place-holders for the results of database accesses. A typical query might be to locate the project entity with a given primary key, or to extract the telephone number of the contact person for a project. The printing application submits each query to the query engine and uses the results to expand the template.

There is a single template for each class of document. Thus all project hand-outs contain the same information in the same format, although the hypertext project descriptions show some individual variation.

We chose to prototype our system in Caml Light[Cousineau90], a dialect of the ML language. This has proved to be an excellent platform for developing "proof of concept" tools, and we have extended the basic system with a small library[Dobson94b] of types and functions encapsulating the common WWW operations. The lightweight database interpreter and query engine form part of this library, making it easy to develop additional database-aware tools.

The printing application may be accessed by adding a "print" icon to a page which links to a CGI script invoking the flattening system – the rightmost icon at the bottom of figure 1 generates a flattened copy of the project description with a single click.

The result of the flattening process is an HTML document – usually, though not necessarily, without hyperlinks – which may then be printed directly from a browser or passed to some intermediate rendering system for further processing (especially useful for high-quality copy).

Spanning Servers

Entities are related using REL elements, which may contain an HREF attribute pointing to the document containing the target entity, This attribute allows an application to follow information using the database schema rather than the hyperlinks. Typically the hyperstructure of a document will be richer than the relationship structure of the lightweight database, so this may reduce the amount of traversal necessary.

If two servers share a common schema for (parts of) their information, they may include relationships to each others' entities. This allows the servers to co-ordinate their information into a single virtual database. A sample application might be where two sites co-operate in providing a bibliography of published work. They may agree on a common schema for references, and mark-up their reference lists accordingly. Such schemata have been developed for a number of purposes by other projects[Genesereth90]. There is however a real danger that different sites might adopt subtly different models for the same information, resulting in a proliferation of non-interworking lightweight databases.

A similar approach may be used to generate indices to information using "robots" – autonomous daemons which scavenge for information in WWW. By publishing the semantic content of pages, sites offer better search and retrieval possibilities than simple keyword searches, and can avoid the overhead involved in speculative traversal of links by "data miner" applications.

In general a lightweight database is equally accessible to client- or server-side processing, which means that processing may occur wherever is most efficient or most convenient.

Related Work

Integrated Relational Databases

As mentioned above, several authors (*e.g.* [Varela94]) have reported integrating database engines to WWW. A typical example is a site's telephone directory, which is accessed using a CGI script interfacing to a relational database engine such as Ingres.

We see the present work as being complementary to these efforts. A lightweight database will never be a substitute for a full RDBMS in applications with substantial searching requirements. However many applications have substantially less intensive requirements. Our example of structuring a document is typical: few sites would store their documents in an RDBMS as a matter of course, but turning a document into a lightweight database facilitates searching and re-formatting with little effort.

A further feature of our approach is that it makes the database public. The mark-up essentially publishes the structure of the information, rather than having it tied-up in a package. This makes client-side querying – as opposed to keyword-only searching – a real possibility, off-loading processing from the central server[DeBra94].

SGML and Hytime

Equally important is the migration of WWW towards the SGML and Hytime standards. These will allow WWW to offer richer and more varied mark-up styles, possibly tailoring the tags used towards exactly the sort of machine-readability we have been investigating with the current work.

We believe that our approach allows simple, lightweight, generic databases to be constructed within web pages. These databases may use any appropriate data model agreed between communicating parties. The extra elements are sufficiently simple that they do not enlarge documents so as to impact on performance, and may easily be recognised and stripped from documents if required.

SGML offers the possibility of alternative, document-specific mark-up to define the structure of information. This allows more "targeted" mark-up. It also allows documents which do not have a clear relational structure to have semantics added to them – clearly a problem in our approach. There is a significant cost to moving to full SGML, however, and it is by no means clear how far WWW will evolve in this direction.

We believe a more important point is that our approach couples the semantics of a page directly with its text, making it difficult to support multiple views onto a set of pages. In effect this is a similar problem to that of explicit link and anchor tags in HTML – the author pre-defines the hyperlink (or database) structure, limiting future re-use and expansion. However the same techniques which decouple links from documents in Hytime or Microcosm[David93] may also be employed in lightweight databases.

The decision to move towards machine- as well as human-readable documents is the important point we wish to stress. The exact approach taken is in many respects far less important than the results of making the web more accessible to automated processing.

Conclusion

We have presented a small extension to HTML to include semantic information into a document's mark-up. The semantics follow an underlying schema based on a well-known database formalism, and allows the construction of generic "lightweight" databases which may span servers. The mark-up may be used by client- or server-side applications to extract information directly from web pages using relational queries.

We have presented an example of generating printed copies of hypertext documents, using mark-up to extract sections for printing. This avoids the need to follow hyperlinks, and allows alternative sections to be inserted or omitted as required.

Further work will concentrate on making queries more efficient, checking the integrity of a lightweight database against a conceptual model, and exchanging data between lightweight databases and full RDBMS systems.

We believe that information retrieval using automated tools is a valuable way of reducing the information overload in WWW. Making the semantic content of pages available directly improves the accessibility of the information and – by allowing improved searching and indexing – may reduce the need for keyword-only searches.

Acknowledgements

The initial impetus for this work came from Keith Jeffery and Eric Thomas. Keith also led us to consider many points which we would otherwise have missed. Wernher Behrendt made many helpful comments in the development stages, and Roger May coached us in the intricacies of CGI.

References

[HTML94]
Tim Berners-Lee and Daniel Connolly, *Hypertext markup language specification – 2.0*, IETF HTML Working Group (November 1994).
http://www.ics.uci.edu/pub/ietf/html/html2/

[Cousineau90]
Guy Cousineau and Gérard Huet, *The Caml primer*, Rapport 122, INRIA Rocquencourt (1990).

[Date86]
CJ Date, *An introduction to database systems, volume 1,* Addison Wesley (Fourth edition 1986).

[David93]
H David, W Hall, A Pickering and R Wilkins, *Microcosm: an open hypermedia system*, Proceedings of INTERCHI '93.

[DeBra94]
PME De Bra and RDJ Post, *Information retrieval in the World-Wide Web: making client-based searching feasible*, Proceedings of the 2nd International World Wide Web Conference (1994).
Accessible *via* http://www1.cern.ch/WWW94/PrelimProcs.html

[Dobson94a]
Simon Dobson and Victoria Burrill, *Lightweight data mark-up*, WWW/04/94, Informatics Department, Rutherford Appleton Laboratory (November 1994). http://www.cis.rl.ac.uk/proj/www/docs/lightweight/

[Dobson94b]
Simon Dobson, *The mlweb library*, WWW/06/94, Informatics Department, Rutherford Appleton Laboratory (November 1994).
http://www.cis.rl.ac.uk/proj/www/docs/mlweb.ps

[Genesereth92]
Michael Genesereth and Richard Fikes (eds), *Knowledge Interchange Format v.3.0 reference manual*, Computer Science Department, Stanford University (June 1992).

[SoftQuad]
SoftQuad Inc., *The SGML Primer*.

[Varela94]
Carlos A. Varela and Caroline C. Hayes, *Zelig: schema-based generation of soft WWW database applications*, Proceedings of the 2nd International World Wide Web Conference (1994).
Accessible *via* http://www1.cern.ch/WWW94/PrelimProcs.html