Web-Site Management: The Strudel Approach

Mary Fernandez — AT&T Labs — mff@research.att.com

Daniela Florescu — INRIA Roquencourt — dana@rodin.inria.fr

Alon Levy — Univ. of Washington — alon@cs.washington.edu

Dan Suciu — AT&T Labs — suciu@research.att.com

1 Introduction

The World-Wide Web (WWW) has become a prime vehicle for disseminating information. As a result, the number of large Web sites with complex structure and that serve information derived from multiple data sources is increasing. Managing the *content* and the *structure* of such Web sites presents a novel data management problem, which has not been previously considered by the database community.

To understand the problem, consider a Web-site builder's tasks: (1) choosing and accessing the data that will be displayed at the site, (2) designing the site's structure, i.e., specifying the data contained within each page and the links between pages, and (3) designing the visual presentation of pages. In existing Web-site management tools, these tasks are, for the most part, interdependent. Without any site-creation tools, a site builder writes HTML files by hand or writes programs to produce them and must focus simultaneously on a page's content, its relationship to other pages, and its visual presentation. As a result, several important tasks, such as automatically updating a site, restructuring a site, or enforcing integrity constraints on a site's structure, are tedious to perform. To support these tasks naturally, we view the problem from a data management perspective.

We decided to study the web-site construction and management problem from a database perspective. We have developed the STRUDEL system [6, 7], which applies concepts from database management systems to Web-site creation and management. In particular, STRUDEL supports declarative specification of a Web site's content and structure and automatically generates a browsable Web site from a specification. STRUDEL's key idea is separating the management of a Web site's data, the management of the site's structure, and the visual presentation of the site's pages.

Broadly speaking, using STRUDEL, the site builder first creates an integrated view of the data that will be available at the site. The Web site's raw data resides either in external sources (e.g., databases, structured files) or in STRUDEL's internal data repository. In STRUDEL's mediator component, as in all of its other components, all external or internal data is modeled as a labeled directed graph, which is the model commonly used for semistructured data [1, 4]. A set of source-specific wrappers translates the external representation into the graph model. The integrated view of the data is called the *data graph*. Second, the site builder declaratively specifies the Web site's structure using a *site-definition query* in STRUQL, STRUDEL's query language. The result of evaluating the site-definition query on the data graph is a *site graph*, which models both the site's content and structure. Third, the builder specifies the visual presentation of pages in STRUDEL's HTML-template language. The HTML generator produces HTML text for every node in the site graph from a corresponding HTML template; the result is the browsable Web site.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

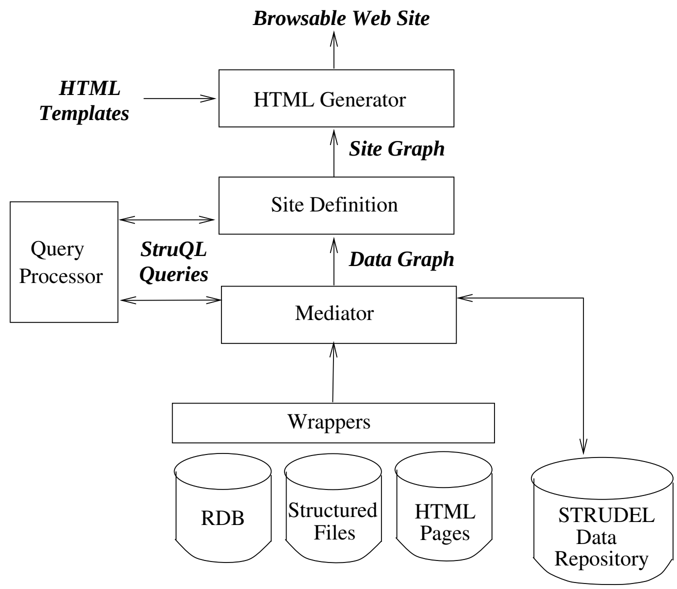

Figure 1: STRUDEL Architecture

STRUDEL is based on a semistructured data model of labeled, directed graphs. This model was introduced to manage *semistructured* data, which is characterized as having few type constraints, irregular structure, and rapidly evolving or missing schema [1, 4]. This data model was appealing for STRUDEL, because Web sites are graphs with irregular structure and non-traditional schemas. Furthermore, semistructured data facilitates integration of data from multiple, non-traditional sources.

As it stands, STRUDEL provides several benefits. Since a Web site's structure and content are defined declaratively by a query, not procedurally by a program, it is easy to create multiple *versions* of a site. For example, it is possible to build internal and external views of an organization's site or to build sites tailored to novice or expert users. Currently, creating multiple versions requires writing multiple sets of programs or manually creating different sets of HTML files. In STRUDEL, a site builder produces multiple sites by applying different site-definition queries to the same underlying data or by creating multiple visual presentations for the same site graph. STRUDEL's architecture also supports evolution of a Web site's structure. For example, to reorganize pages based on frequent usage patterns or to extend the site's content, we simply rewrite the site-definition query. Declarative specification of Web sites can offer other advantages. For example, it becomes possible to express and enforce integrity constraints on the site and to update a site incrementally when changes occur in the underlying data.

STRUDEL clearly separates the three tasks of building Web sites and is the first system that supports declarative specification of a site's content *and* structure. Other recent research prototypes have discussed the separation of the three tasks, but do not not support declarative specification of content or structure [3, 8]. Other research projects support declarative specification, but merge the tasks [2, 5]. Commercial tools such as Vignette's StoryServer and those provided by major database vendors separate the management of the underlying data from its visual presentation. Individual pages or sets of related pages are constructed dynamically by evaluating queries that are embedded in HTML templates; query results are merged into HTML templates to produce pages. Other products provide graphical user interfaces that support drag-and-drop editing of individual pages (e.g., Microsoft's FrontPage, NetObjects' Fusion) or of the structure between individual pages (e.g., Elemental's Drumbeat).

This intense activity in research and industry indicates that Web-site management is an important problem, and because its central issue is management of site content and structure, it should be of interest to the database community. Given this, we set out to gain experience quickly using STRUDEL so that we may understand which aspects of Web-site management benefit most from application of database concepts and identify the critical research issues we should focus on in this realm. In this paper, we briefly describe the STRUDEL system, the lessons we learned from applying the system, and outline the directions we consider important for future research.

2 The Strudel System

Strudel's architecture is depicted in Fig. 1; rectangles depict processes and emboldened terms specify the inputs and outputs of the processes.

We describe the architecture bottom up from the site builder's perspective. Using Strudel, the site builder first creates an integrated view of the data that will be available at the site. The Web site's raw data resides either in external sources (e.g., databases or structured files such as Word or Excel documents) or in Strudel's internal data repository. In Strudel, all external and internal data is modeled as a labeled directed graph. In this model, the database consists of objects connected by directed edges labeled with string-valued attribute names. Objects are either nodes, identified by a unique object identifier or are atomic values, such as integers, strings, and files. Objects are grouped into named collections, which are used in queries.

A set of source-specific wrappers translates the external representation into Strudel's graph model. The site builder can produce an integrated view of several sources by writing a query in StruQL, Strudel's query language. The integrated view of the data is called the data graph. Next, the site builder declaratively specifies the Web site's structure using a site-definition query also in StruQL. The result of evaluating the site-definition query on the data graph is a site graph, which models both the site's content and structure. Third, the builder specifies the visual presentation of pages in Strudel's HTML-template language. The HTML generator produces HTML text for every node in the site graph from a corresponding HTML template; the result is the browsable Web site.

Strudel provides several benefits. Since a Web site's structure and content are defined declaratively by a query, not procedurally by a program, it is easy to create multiple versions of a site. For example, it is possible to build internal and external views of an organization's site or to build sites tailored to novice or expert users. Currently, creating multiple versions requires writing multiple sets of programs or manually creating different sets of HTML files. In Strudel, a site builder produces multiple sites by applying different site-definition queries to the same underlying data or by creating multiple HTML renderings of the same site graph. Strudel's architecture also supports evolution of a Web site's structure. For example, to reorganize pages based on frequent usage patterns or to extend the site's content, we simply rewrite the site-definition query.

2.1 Example Web Site

The following example shows how one author's homepage site is generated by Strudel. The main data source is the author's Bibtex bibliography file. The homepage site has four types of pages: the root page containing general information, a page containing all paper abstracts, and pages containing summaries of papers published in a particular year or category. We describe the first two steps of the site-definition process: creating the data graph from a Bibtex file and defining the site graph in StruQL.

Fig. 2 contains a fragment of the site's data graph and was generated by a Bibtex wrapper; the wrapper converts Bibtex files into a Strudel data graph. Both objects, pub1 and pub2, are members of the Publications collection. It is important to note here that data in Strudel's graph data model does not have a fixed schema, which facilitates integration of data from multiple sources; for example, objects in the same collection may have different representations as do pub1 and pub2 below.

The homepages' site graph is defined by the query in Fig. 3. The first clause creates two new objects called RootPage and AbstractsPage and creates a link between them. The second clause (lines 6-7) creates two new objects, AbstractPage(x) and PaperPresentation(x) for each member x of the Publications collection; these objects contain the publication's information that will appear in different parts of the site. For example, the expressions on lines 9-10 copy all of x's attributes and values into the new objects. The link clause also encodes inter-page

```
object pub1 in Publications {
  title "Specifying Representations..."
  author "Norman Ramsey"
  author "Mary Fernandez"
  year 1997
  month "May"
  journal "Transactions on Programming..."
  pub-type "article"
  abstract is text "abstracts/toplas97.txt"
  postscript is ps "papers/toplas97.ps.gz"
  volume "19 (3)"
  category "Architecture Specifications"
  category "Programming Languages"
}
```

```
object pub2 in Publications {
  title "Optimizing Regular..."
  author "Mary Fernandez"
  author "Dan Suciu"
  year 1998
  booktitle "Proc.  of ICDE"
  pub-type "inproceedings"
  abstract is text "abstracts/icde98.txt"
  postscript is ps "papers/icde98.ps.gz"
  category "Semistructured Data"
  category "Programming Languages"
}
```

Figure 2: Fragment of data graph for example site

```
// Create Root & Abstracts page and link them
CREATE RootPage(), AbstractsPage()
LINK RootPage()−>”AbstractsPage”−>AbstractsPage()

// Create a presentation for every publication x
WHERE Publications(x), x−>l−>v
CREATE PaperPresentation(x), AbstractPage(x)
LINK
  AbstractPage(x) −> l −> v,
  PaperPresentation(x) −> l −> v,
  PaperPresentation(x)−>”Abstract”−>AbstractPage(x),
  AbstractsPage() −>”Abstract” −> AbstractPage(x)

{ // Create a page for every year
  WHERE l = ”year”
  CREATE YearPage(v)
  LINK
    YearPage(v) −> ”Year” −> v
    YearPage(v)−>”Paper”−>PaperPresentation(x),

    // Link root page to each year page
    RootPage() −> ”YearPage” −> YearPage(v)
}

{ // Create a page for every category
  WHERE l = ”category”
  CREATE CategoryPage(v)
  LINK
    CategoryPage(v) −> ”Name” −> v,
    CategoryPage(v)−>”Paper”−>PaperPresentation(x),

    // Link root page to each category page
    RootPage() −> ”CategoryPage” −> CategoryPage(v)
}
```

Figure 3: Site definition query for example homepage site

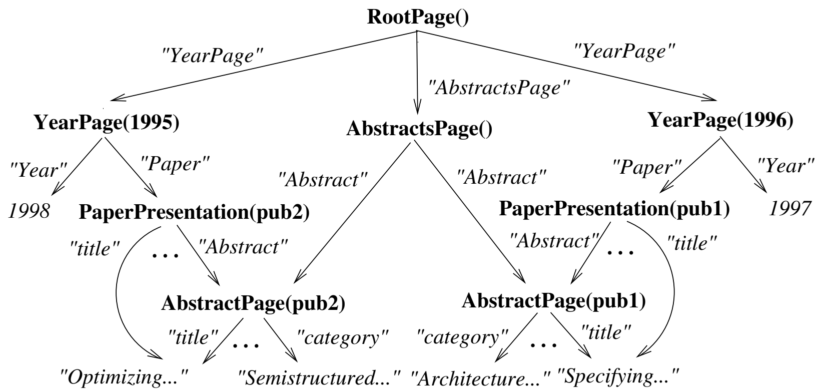

Figure 4: Fragment of site graph for example homepage site

structure. On line 12, the general abstracts page is linked to the abstract page of each publication (AbstractPage(x)). The nested where clause (lines 14-23) creates a page for each year associated with a publication; the link clause associates each PaperPresentation object with its corresponding YearPage. Lastly, the root page is linked to each year page. A similar clause creates a page for each publication category and links category pages to PaperPresentation objects. Note that the site-definition query only specifies what information will be available in the site and what relationships exist between pages in the site; it does not specify anything about the site's visual presentation. The result of applying a site-definition query to a data graph is another graph in Strudel's data model; this permits us to create site by composing multiple queries.

Fig. 4 depicts a fragment of the generated site graph; for clarity, it excludes the result of the last nested clause that produces category pages. Note that the site graph encodes both the site's content and its structure. For example, the PaperPresentation objects have links to paper titles and to their associated abstract pages. All leaf objects contain page content, e.g., the titles of publications. Declarative specification of the site graph is powerful, because the site builder can specify its structure in any order he chooses. For example, he can define the pages "top down" from the root, or first define each group of related pages and then link them.

3 Experiences with Strudel

After implementing STRUDEL's first prototype, we wanted to evaluate STRUDEL's methodology and our choice of the semistructured data model. First, we considered whether our premise that the three tasks of Web-site creation can and should be separated holds in practice. Specifically,

- Is there always a clear separation between these tasks? If not, in which cases do their mutual dependencies make separating them counter productive?
- For what kinds of Web sites is STRUDEL most effective? How useful is the ability to explicitly and declaratively manage a Web site's structure?

We have had both practical and exploratory experiences with STRUDEL. In our practical experience, we used the STRUDEL prototype to create sites for individual users and for two organizations (AT&T Research being the largest one), and to create a version of the CNN Web site for demonstration purposes. In our exploratory experience, we described our methodology and demonstrated our prototype to several potential STRUDEL users.

We learned several important lessons concerning STRUDEL's methodology from our study. Separating the management of the underlying data from other Web-management tasks is the basis for several commercial products, e.g., most commercial relational and object-oriented databases provide Web interfaces to their systems.

STRUDEL provides two other important features: the abilities to integrate data from *multiple* sources and to incorporate unstructured sources (e.g., structured files). The AT&T Research site, for example, integrated five data sources.

Isolating the management of a site's structure was also important. For example, CNN's Web-site group is building a specialized tool for managing site structure. We also found that building complex Web sites is an iterative process in which the site structure evolves over time. For example, creating the AT&T and Rodin sites required several iterations. Declarative specification of the site's structure enables easy changes to a site. Finally, STRUDEL is most effective when multiple versions of a site are built from the same underlying data. For instance, once we built AT&T's internal research site, building the external version was trivial.

Separating management of the site's structure and its visual presentation is more subtle. This separation simplifies creating multiple versions of a site especially when the site's structure is the same in all versions, but its visual presentations differ. In this case, all versions share one site graph, but each version has its own HTML templates. It is not always clear, however, which aspects of a site should be encoded as structure or as visual presentation. For example, the AT&T external site is derived from the internal site by excluding the attributes of some objects in the generated pages; in this case, it is easier to create HTML templates that omit these attributes than it is to create a new site graph that explicitly excludes those attributes. Consider the order of articles or the placement of images in a page at the CNN site. Such information could be encoded in the visual presentation or in the site's structure. For CNN, managing this information is crucial, because they consider these editorial elements a primary value of their site.

To characterize the sites for which STRUDEL is most useful, we consider two criteria: the amount of data they contain and their structural complexity. Measuring the amount of data in a site is straightforward. One possible measure of structural complexity is the number of `link` clauses in the site-definition query. In current practice, an analagous measure of site complexity is the number of CGI-BIN scripts required to generate a site.

We observed that STRUDEL is most useful for sites that have complex structure and whose structure is dependent on the underlying data. For example, the CNN Web site contains a large number of articles. Although the disposition of an article in a site is complex (i.e., it appears in several formats on different pages and is linked to many other pages), the structure is uniform for all articles in the site. This uniformity also applies to all people in the AT&T site and all publications in the example homepage sites. In summary, when a site has simple structure and little data WSYWIG tools such as Microsoft FrontPage or NetObjects Fusion are appropriate choices. When a site contains large amounts of data, but has simple structure, then a tool that provides a Web-based interface to a database is appropriate. However, when there is a lot of data and the site structure is complex, STRUDEL is most appropriate.

4 Future Research

Our experience helped us identify research problems of practical and theoretical interest, which we outline here. They address issues of STRUDEL's applicability to dynamically generated sites, its scalability to larger sites, its usability as an end-user tool, and its interoperability with existing tools.

Incremental generation of the Web site: In STRUDEL's prototype, we precompute a Web site by completely materializing its site graph. Many Web sites, however, cannot be precomputed, because they depend on user input that is not available statically or because the underlying data sources are too large. Currently, STRUDEL does not support dynamically generated sites. In practice, dynamic generation is supported by often large groups of loosely related CGI programs. Supporting dynamic evaluation would eliminate writing such programs by hand.

Although we can decompose a site-definition query into multiple, dynamic queries, and we have theoretical techniques for optimizing these queries, implementing dynamic evaluation requires significant systems-design effort. For example, our optimization techniques cache query results to reduce "click time" for future queries;

these results essentially encode state required by the STRUDEL query processor. An open problem is how and where this state should be stored: in a client-side browser and/or a server-side query processor.

Graphical interface to the query language: Not surprisingly, many potential users of STRUDEL asked whether we can provide a friendly visual interface for specifying queries, instead of having to write STRUQL queries by hand. Clearly, a better interface is needed, probably in the spirit of Query By Example.

Integration with the programming environment: Many commercial tools exist for Web-site creation and management. We do not presume that STRUDEL will replace *all* of them, therefore an important practical issue is how to integrate STRUDEL with existing tools. In particular, developing the appropriate API to STRUDEL may be the best way to incorporate it into tools that Web-site builders currently use.

References

[1] S. Abiteboul. Querying semi-structured data. In *Proceedings of the ICDT*, 1997.

[2] G. Arocena and A. Mendelzon. Weboql: Restructuring documents, database and webs. In *Proceedings of International Conference on Data Engineering*, pages 24–33, 1998.

[3] P. Atzeni, G. Mecca, and P. Merialdo. To weave the web. In *Proceedings of VLDB*, 1997.

[4] P. Buneman. Semistructured data. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Tucson, Arizona*, pages 117–121, 1997.

[5] S. Cluet, C. Delobel, J. Simeon, and K. Smaga. Your mediators need data conversion. In *Proceedings of SIGMOD*, 1998.

[6] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. System demonstration - strudel: A web-site management system. In *ACM SIGMOD Conference on Management of Data*, 1997.

[7] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the boat with strudel: experience with a web-site management system. In *Proceedings of SIGMOD*, 1998.

[8] P. Paolini and P. Fraternali. A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. In *Proceedings of EDBT Conference*, Valencia, Spain, 1998.