Integrating Town Information Servers

Lukas C. Faulstich*

Institut für Informatik, Freie Universität Berlin,
<http://www.inf.fu-berlin.de/~faulstic>

Abstract

This article presents a case study on the integration of Web sites based on the HyperView approach. The design of a prototype for a virtual Web site is described in order to demonstrate the application of the HyperView approach in the field of cultural town information. A number of open issues arising from this particular application are identified and discussed.

1 Introduction

The main problem with today's WWW is not a lack of information, but the diversity of sources on which the information is dispersed. To use the WWW for planning, even for simple tasks – like for instance to plan an evening in a city with rich cultural life like Berlin – often means to find and browse a number of different specialized Web sites in order to get an sufficiently complete picture for making a decision. Hence there is an increasing need for integration of WWW sources.

The integration of databases is a well-studied field. Unfortunately, database integration techniques do not apply directly to WWW data integration since WWW sources are lacking essential database features like query processing or schemata. The research area on semistructured data ([Abi97, Bun97]) deals with approaches to adapt well-known database techniques like queries, views, and schemata to semistructured data. In particular, HTML pages and XML documents in the Web are often semistructured in the sense that groups of structurally similar documents can be identified. The existence of such regularities in Web sites makes it feasible to write wrappers for extracting data from these Web sites.

The goal is to set up *virtual Web sites* that integrate data from a number of source Web sites. Virtual Web sites appear to the user just like conventional Web sites. The only difference is that the information presented to the user forms a view of the underlying Web sites from which data is extracted on demand. Virtual Web sites hide the structural heterogeneity of their sources, provide missing cross-links and achieve a broader coverage. The HyperView approach to the integration of Web sites [Fau99b, Fau99c, FS99] provides a methodology, a formal foundation, and a software platform for setting up such virtual Web sites.

1.1 Town Information

We define "Town Information" as information related to events, infrastructure and services of some urban community. In particular we focus on cultural and entertainment related information. Hence,

*Supported by the German Research Society, Berlin-Brandenburg Graduate School on Distributed Information Systems (DFG grant no. GRK 316)

events describe concerts, movie shows, street markets and other more, infrastructure means "locations" of events, i.e., theaters, concert halls as well as restaurants, pubs, bars, etc. Services include public transportation (e.g., bus and subway schedules), ticket reservation, weather forecasting, shopping, and so on.

To a large extent, town information is already available at various Web sites. Although a few servers of city magazines provide ample data on this domain, it is still necessary to consult other servers for additional information. For instance, detailed concert announcements can often only be found at the Web sites of the concert halls or clubs, e.g., of the jazz club "A-Trane"[1] which is presented in this chapter. Also, town maps and public transportation information are typically available from dedicated servers only.

Using the sources mentioned above, it is feasible to plan for instance how to spend an evening in Berlin. However, when trying to find and combine relevant pieces of information, the user is confronted with various problems. The main problem is the *missing link*: related data on different servers are not connected by hyper-links. If the user tries to find related information by consulting another server, he often encounters *semantic heterogeneity* that can make data from one server useless in querying another server. For instance, arbitrary addresses can not be used in the search form of the BVG public transportation server which requires the user to enter names of bus stops.

Another problem is the *incomplete coverage* of single servers. For instance, there is no server that reliably lists *all* concerts in Berlin. To get the full picture, information has to be collected from several Web Sites.

Updates in the sources are another problem. To become aware of changes, the user has to check relevant servers on a regular basis. Finally, new Web sites are set up and existing ones are closed down quite often. This *source fluctuation* makes it necessary to discover new sources and find replacements for those that have stopped operation.

Using the HyperView System to integrate town information from different servers, we can add *missing links* and achieve a more *complete coverage*. The problem of semantically incompatible information in different Web servers cannot be solved by HyperView per se, it only provides the platform to use some conversion service to map information from one Web Site to the terminology required by some other Web Site. Such a conversion service can be some external Web Site, some mapping rules implemented by the HyperView designer, or an internal data base or thesaurus maintained by the HyperView administrator.

The problem of *source updates* can be solved by implementing notification services on top of the database layer of HyperView. Without such a notification service, the HyperView System at least partially automates a search for information and thus makes it easier to repeat it when the information is actually needed.

The problem of data changes is addressed by using the HyperView database primarily as a cache which can be cleared after a source dependent period of time. The rule concept of HyperView aims at making the system robust against small structural changes in the sources and at facilitating the rapid specification of ACR-views over HTML pages.

1.2 Organization of the paper

The rest of the paper is organized as follows: In Section 2, an overview on the HyperView approach to the integration of Web sites is given. An scenario for our case study on Berlin town information is presented in Section 3. In Section 4 the development of schemata and views for building an integrated

[1]<http://www.a-trane.de/>

event calendar using HyperView is demonstrated with examples. Section 5 reviews related work on Web site integration and Section 6 concludes the paper with a discussion of open problems.

2 The HyperView approach

To solve the problem of integration of semistructured data sources as outlined above, three subproblems have to solved:

Extraction: relevant data has to be extracted from the sources. For sources being Web sites, this means loading HTML pages, finding the relevant pieces of information on these pages, and identifying the relations between them. The requirement that sources are well-known and sufficiently regular allows to define the extraction methods in advance.

Integration: data from different sources and from different locations within a source has to be combined and restructured into a uniform semantic representation. This step relies on the existence of a global schema of reasonable size due to a narrow application domain.

Presentation: the user interface supports the user in formulating and submitting information requests and presents responses consisting of parts of the integrated data in a certain target format. The primary target format is dynamically generated HTML, because it allows to set up a Web site which the user can browse using a standard HTTP client to access the integrated data.

These steps are reflected in the HyperView methodology and architecture. HyperView offers an unifying formal model that applies to all three of these steps and a generic system based on this formalism that implements virtual Web sites defined by declarative specifications for extraction, integration, and presentation.

2.1 Data Model and View Mechanism

The three steps introduced above can be conceived as consecutive mappings. Each of this mappings can be implemented as a view on its input data that provides the input data for the next step.

To reduce the complexity of the system, a uniform data model and view mechanism is to be used throughout all steps. A graph-based data model has been chosen since graphs have been found suitable for representing semistructured data [Abi97, QWG$^+$96, BDHS96, FFLS97]. Thus directed labeled graphs are used uniformally to represent the input data (HTML pages), the data extracted from these pages, the integrated data, and finally the data returned to the user. Graphs are described by graph schemata which are again graphs. Since we have to handle several graphs at a time, we have developed the *Clustered Graph Data Model* (CGDM). In this model, each graph in the system is viewed as a so-called *cluster* of a global modularized graph. Using this data model, the WWW is represented conceptually as a clustered graph where each HTML page forms a cluster of its own. Hyper-links between different are represented by inter-cluster edges.

The view mechanism needed in HyperView has to support mappings between graphs. To be more precise, it must transform of one or several input graph clusters into an output graph cluster. Since space and time limitations as well as source updates make it difficult to extract and integrate all available information in advance, the view mechanism must support incremental materialization in response to user requests. Graph transformation techniques have been intensively studied in the field of graph grammars (cf. [Roz97]). However, the requirements of modularization and of demand driven execution are not met by the existing approaches. Therefore we have developed a special kind of graph-transformation rules by which views on graphs can be materialized using demand-driven activation strategy for rules.

To support the composition of views, rule activation propagates through several layers of views in a backward-chaining fashion. By this, queries expressed as requests for graph elements are propagated to the bottom layer. In the HyperView System where the bottom layer models the WWW, this layer is materialized on demand by loading the requested pages and converting them into new graph clusters. The data added to each layer allows the rules of the succeeding view to be applied and to extend the next layer. Thus, a cascade of rule activations finally propagates the requested information back to the top layer.

This novel view mechanism of HyperView forms a key contribution of this work. The formal treatment of the *Clustered Graph Data Model* (CGDM) and of the view mechanism is presented in [Fau99a].

2.2 Architecture

In order to separate the steps of information extraction, integration, and presentation, we use a hierarchy of views which map information from a lower to a higher level of abstraction. These levels of abstraction are reflected by the layers of the conceptual HyperView architecture depicted in Figure 1.

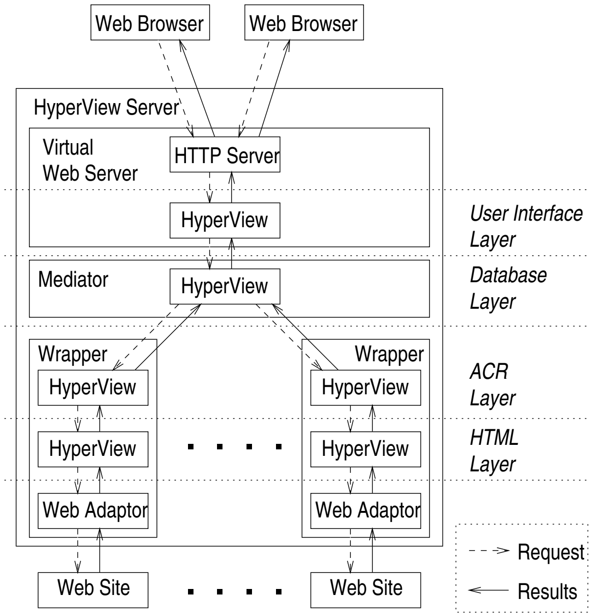

Figure 1: The HyperView architecture.

The HTML layer models single HTML pages loaded from the Web sites at a purely syntactical level. The *Abstract Context Representation* (ACR) layer represents the data extracted from each Web site. This data is described by a schema, the *ACR schema* of the site. It models the relevant content of the Web site at a semantic level, but organized according to the Web site's structure. The ACR schemata are analogous to the component *export* schemata in the well-known reference architecture of [SL90] for federated database management systems. Note, that ACR schemata do not correspond to the *internal* component schemata since they capture only the data relevant for the integration.

Information extracted from different Web sites is combined at the database (DB) layer which provides an integrated domain-oriented view on top of the ACR layer. The DB layer is described by a schema called the *DB schema*. This schema corresponds to the *federated schema* in [SL90].

Finally, the user interface (UI) layer consists of a set of views on the database layer which are then formatted as HTML pages and delivered to the user's HTTP client. Each of these views is described by a corresponding UI schema. In [SL90], these schemata are analogous to *external schemata*.

Besides virtual Web sites, all other conceivable kinds of applications that rely on integrated information from Web sites can be built on top of the database layer. One example of such an application in the field of digital libraries is presented in [FS99].

From the viewpoint of systems design, the layered conceptual architecture of HyperView fits well in a wrapper-mediator architecture [Wie92]. Each Web site can be encapsulated into a site-specific wrapper which implements the HTML and the ACR layer. Information from different wrappers is integrated in a mediator which implements the DB layer. The UI layer is implemented by the user interface which can be seen as a specialized kind of mediator.

The basic building block of the HyperView architecture is the component depicted in Figure 2. Each HyperView components implements a single view. It contains a processor that receives HVQL queries against the graph base of the component. This graph stores the already materialized part of the view. If the query cannot be answered from the current contents of the graph base, the processor uses view definitions stored in a rule base to query other HyperView components and to materialize the needed graph elements on the fly.

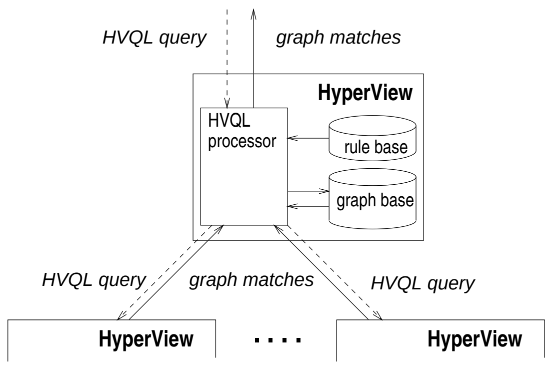

Figure 2: The basic building block of the HyperView architecture.

The bottom most HyperView components are called *Web Adaptors* since they do not issue HVQL queries, but load pages from the underlying Web sites in order to answer the queries they have received.

2.3 Usage Scenario

In setting up and using a virtual Web site based on HyperView, we can distinguish the following four roles:

the *domain expert* selects the information sources and decides how which information is to be offered to the user as a result of integrating the underlying sources.

the *designer* cooperates with the domain expert to refine his ideas and to implement the schemata and HyperViews needed to achieve the integration proposed by the domain expert. Due to the modular architecture of the HyperView system, several *designers* work independently on different development tasks.

the *administrator* operates the HyperView server. This involves mostly system administration like configuration of the underlying HTTP server, backup of local data etc. Maintenance of HyperView definitions is carried out by the *designer*.

the *user* retrieves information offered by the HyperView server. The *user* can configure the presentation of the retrieved data by changing settings in his/her user profile. There may be several groups of users having different information demand, expertise, or access privileges. HyperView can provide different virtual Web sites to each of these groups.

Depending on the tool support and expertise available, the domain expert and the administrator can act as designer. Moreover, it is conceivable that the user is the domain expert who maintains a HyperView server for his/her personal use. Normally however, a HyperView server will be operated by some organization as a service to some community. This community could for instance be a group of intranet users in a company or a certain scientific community within the internet.

3 Scenario

In this case study we implemented the prototype of an integrated town information server based on the HyperView System. We have included exemplary Web sites that are suitable for demonstrating the potential of such a server and the problems to be solved in practice.

The following list presents the exemplary Web sites that have been analysed for inclusion in the prototype:

- The Web site of the town magazine “Zitty” (<www.zitty.de>) consists of two databases, the “cineProgrammat” for movie theater programs, and the “bplusProgrammat” for all other kinds of cultural events, including concerts, plays, vernisages etc. Both databases offer addresses of locations.
- “Tonwerk” is a music magazine that offers at <www.tonwerk.de> a database of concerts, including occasional links to group/location home pages; its ACR schema is depicted in Figure 5.
- The jazz club “A Trane” offers its monthly program at <www.a-trane.de>. The program includes descriptions and pictures of the groups; cf. Figure 6
- “Kino Berlin” (<www.kino-berlin.de>) is a service that offers movie theater programs for Berlin. The Film database includes short descriptiones and plot summaries. Figure 4 shows its ACR schema.
- The “Internet Movie DataBase” (IMDB, <www.imdb.com>) stores detailed information about almost all movies ever produced. For the purpose of a town information service, links to movies and to collections of external reviews are sufficient. Hence the IMDB is not modeled in great detail.
- “Arsenal” is a cineastic movie theater that maintains a Web site with background information on its monthly program (<http://www.fdk-berlin.de/arsenal.htm>).

- The “Stadtplandienst” <www.stadtplandienst.de> offers town maps of Berlin (and other german cities). It allows to lookup a specific address and to present a detail of the town map corresponding to this address.
- Public transportation: at <http://www.fahrinfo-berlin.de> an online schedule of public transportation in the area of Berlin is available. Using HTML forms, the fastest connection between two bus or train stops can be found. However, no search facility for the bus/train stops closest to some address is provided.

The mentioned Web sites can be combined in the following way:

1. cultural events from city magazines (Zitty cineProgrammat and bplusProgrammat), other general event calendars like Tonwerk, and individual promoters (e.g., A-Trane) are merged into a common cultural event calendar
2. detailed background information on events from location Web Sites (e.g. A-Trane, Arsenal) or genre-specific information services (e.g. IMDB) is associated with the events in the calendar
3. links from addresses of locations to town map details are provided
4. links to home pages of locations, artists, ensembles are added (if available)
5. links from events to the public transportation search form are added with *appropriate presettings* for reaching the respective event

In the resulting virtual Web site, the task of planning how to spend an evening in Berlin could now be performed as follows:

1. Find interesting entertainment events by either stating preferences in a HTML form or browsing the event database by different categories like service, region, venue, time, etc.
2. Follow links to external pages provided by the event promoters for background information.
3. Follow a link from the address of a venue to a detail of the city map showing its surroundings.
4. Search/browse restaurants and pubs by their category and location
5. Query for a public transportation connection between two locations.

4 Developing a cultural event calendar

4.1 Conceptual schema

The main idea for the design of the conceptual schema is that cultural *events* are manifestations of cultural *projects*. A cultural project may be the collection of a museum, a concert program in the repertoire of an orchestra or band, the production of a theater play, a movie etc. Accordingly, an event may describe the exhibition of a collection, a particular concert, the performance of a play, or a movie show at a particular movie theater. The cultural events take place at *locations*. Locations in this sense can be museums, concert halls, movie theaters etc.

Since there are many different arts, projects must be organized into different *genre*s such as music, theater, film etc. For practical purposes this classification is still too coarse. Hence the specialization of genres into subgenres has to be supported. Since this taxonomy must be visible to the user, genres are grouped at instance level into nested collections by their genre. This provides a more flexible data-driven classification scheme than a static a-priori taxonomy.

Since the schema should simplify the navigation for the user, additional edges have to be introduced. For instance, there should be reverse links from events to projects, from projects to genres etc. This can be achieved by simply extending the rules that materialize the forward edges. Furthermore, an event should inherit the location of its project if it is not specified explicitly. Projects belonging to a genre via a *direct_project* edge should belong as well to all more general genres via computed *project* edges. A location should indicate which genres it offers by listing the genres of the projects that have events at that location. These requirements can be satisfied by introducing additional rules that compute these edges.

In Figure 3 the resulting schema for the calendar of cultural events is depicted.

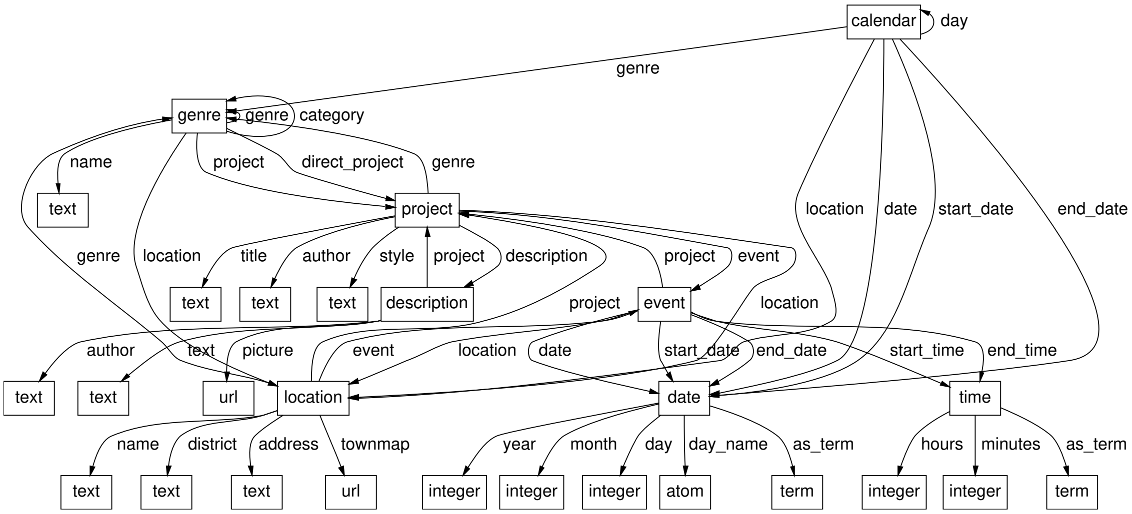

Figure 3: DB Schema of a calendar for cultural events.

4.2 Wrapping town information sites

Wrappers for Tonwerk, A-Trane, and Kino Berlin have been implemented. Their schemata are depicted in Figure 5, Figure 6, and Figure 4, respectively. Moreover, a small wrapper for the IMDB that allows to access IMDB records and links to external reviews for a given movie title has been developed as well. The implementation of wrappers for further Web sites can build on this experience.

The Web site of Kino Berlin (see Figure 4) offers several indices by which the user can find movies and movie theaters. Most relevant for the event calendar is the index of current movies movie_index that is organized by the initials of movie titles (movie_initial nodes). Each movie is presented on a page (movie_page node) that contains a section with information about the movie (movie_info node) and the schedules of the movie at various movie theaters (movie_at_theater node).

The Tonwerk Web site (cf. Figure 5) lists concerts in a different calender for each music style. Each concert is described by the usual attributes, date, time, name of location and band and a short description.

Of the A-Trane Web site, only the program page and its subpages (see Figure 6) are modeled. This page lists concerts. Since some bands may appear on several consecutive days, a date period is

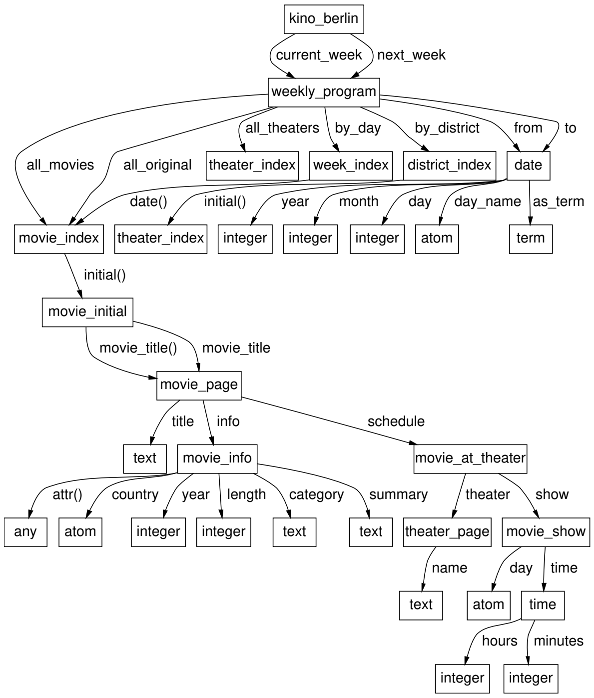

Figure 4: ACR Schema of Kino Berlin.

given for each concert. A long description of the band with pictures is on a separate page modeled by atrane_description nodes.

For each wrapper, two rule sets have been implemented: one that extracts data from the HTML code of Web pages and builds the ACR using this data, and one that uses this ACR to build the calendar database conforming to the conceptual schema presented in Figure 3.

In this subsection we present only a few small examples of relatively simple rules to give an introduction to the view mechanism of HyperView. We use a graphical notation in which new graph elements added by a rule are shown in boldface. Graph clusters are indicated by dashed boxes with round corners. Lowercase node labels indicate corresponding node types in the schema, while variables are denoted by uppercase labels. More complex rules may involve arbitrary constraints on variables and reuse specifications for reusing existing graph elements.

As first example we present rule atrane_program.concert that extracts information about a concert in the A-Trane. This rule is depicted in Figure 7. If a *concert* edge starting from a atrane_program node in the A-Trane ACR is requested, this rule is triggered. It matches a *source* edge pointing to the root node of the HTML page containing the current program of the A-Trane. On this page, it matches rows (tr nodes) of a table listing the concerts. From each row it extracts the text of the cells containing the name of the band and the music style of the band. This information is used to add attributes *group* and *style* to a new atrane_concert node that is connected to the atrane_program node by a new *concert* edge edge.

The information extracted by rule atrane_program.concert is used by rule genre.direct_project to materialize *direct_project* edges for the genre node named jazz. (Since A-Trane is a Jazz club it is assumed that band appearing in this club play Jazz). The rule follows a *atrane_source* edge to the atrane_concert node and the *concert* edge produced by rule atrane_program.concert to

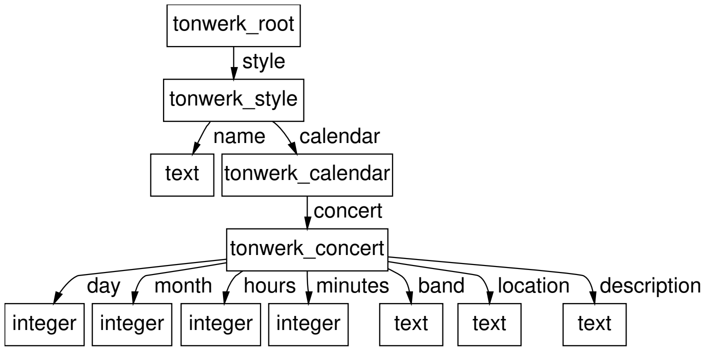

Figure 5: ACR Schema of Tonwerk.

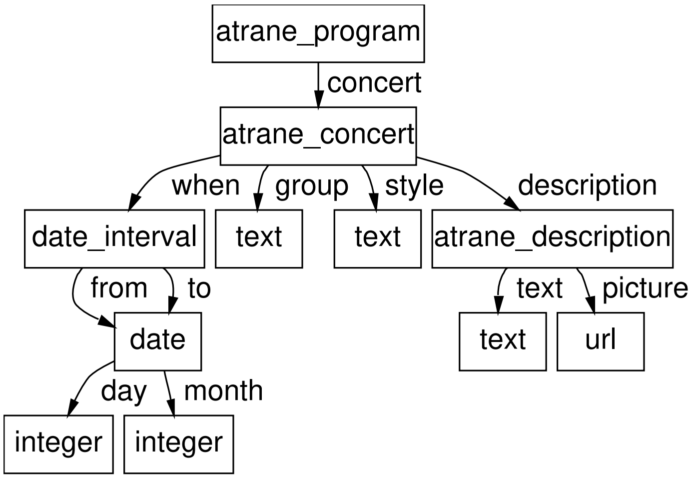

Figure 6: ACR Schema of A-Trane.

match atrane_concert nodes and create for each concert a project node linked to the genre node by a *direct_project* edge. This project node has a *location* edge pointing to the (existing) location node representing the A-Trane. The new *atrane_source* edge pointing to the atrane_concert node is used by other rules to add further data on the concert in the DB layer.

As mentioned above, the calendar graph supports computed *project* edges that point to all projects belonging to a genre *and* to its subgenres. For instance, all jazz projects belong also to the music genre. In Figure 9, rule genre.project that is responsible for materializing *project* edges is depicted. It consists of two alternative rules, the first one matching all projects belonging directly to a genre and the second one recursively matching all projects of subgenres.

4.3 The cultural calendar Web site

In Figure 10, a screen shot depicting the Web interface of the town information prototype is shown. It is based on the generic database browser of the HyperView System which allows the user to explore a HyperView database by browsing the HTML pages generated by the Web interface.

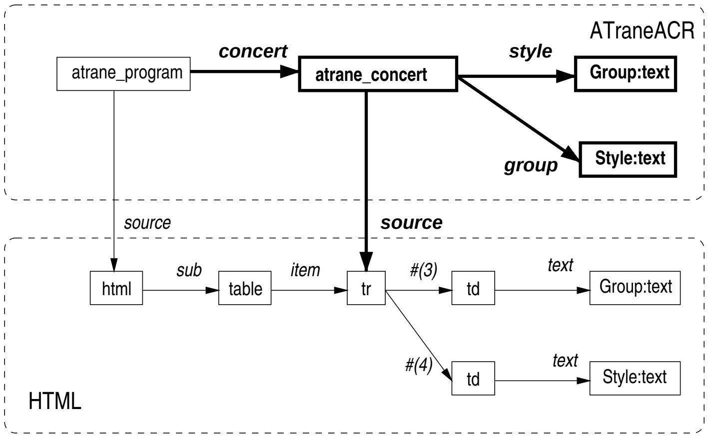

Figure 7: Rule atrane_program.concert.

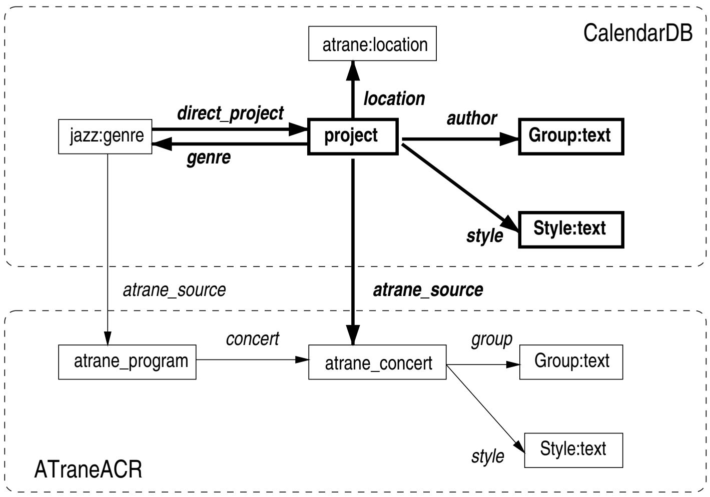

Figure 8: rule genre.direct_project.

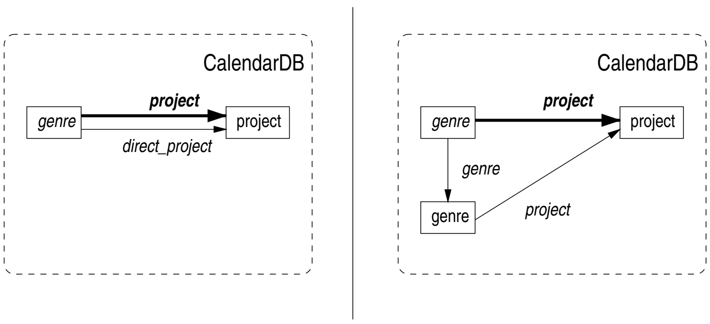

Figure 9: The two alternatives of rule genre.project.

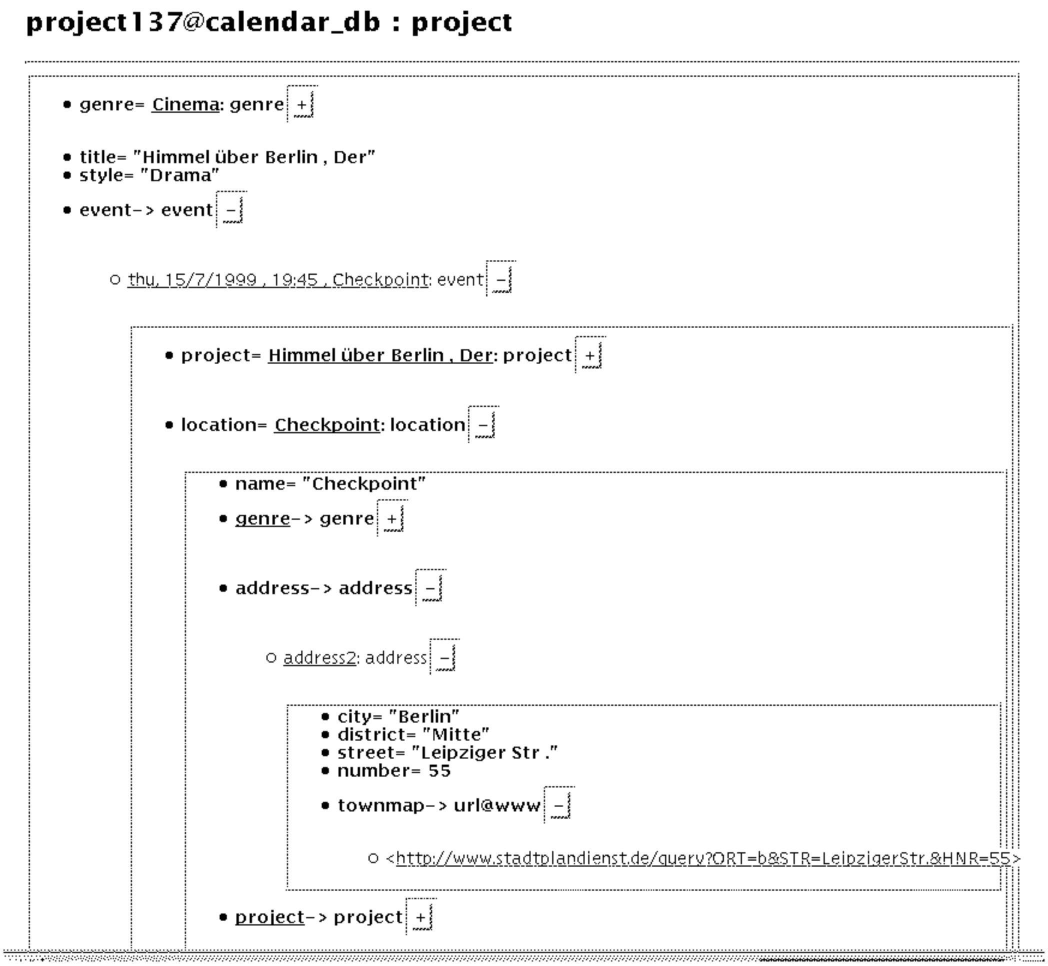

Figure 10: Screenshot of page about the film "Wings of Desire" in the cultural event calendar. For the movie, a set of movie shows is presented as "events", the first of which is expanded. The location of this event, the movie theater "Checkpoint" and its address is also expanded. Note the link from the movie theater's address to the town map detail at the Stadtplandienst server.

5 Related Work

This section gives a brief overview of related work in the field of Web site integration. For each of the steps of data extraction, integration, and presentation there exist various approaches:

Data extraction. Most approaches for data extraction represent Web documents as strings and use regular expression matching to find substrings relevant for extraction. We argue that these approaches neglect important structural information inherent in HTML documents and tend to be not very robust. Therefore we have adopted a graph representation based on HTML parse trees. Recently, this approach has been adopted by other projects like WebOQL [AM98] and W4F [Arn99].

Data integration. Integration of *semistructured* data is often tackled with **query-based approaches**. Web query languages like WebSQL [MMM97] and W3QS [KS97] aim at ad-hoc queries, not at a permanent integration of Web sites. WebOQL is primarily a *query language*, not a data integration tool. It can be used to a limited extent to achieve data integration, but the lacking schema concept seems to make the specification and maintenance of the necessary queries difficult. The same holds for MedMaker [PGMU96] and for Strudel [FFLS97] which do not use any schemata to model input and output data.

In ARANEUS [PA97], the task of integration is left to relational views. However, the large body of relational approaches to data integration does not apply well to semistructured data due to the inherent variability and irregularity of such data.

Rule-based approaches like HyperView, YAT [CDSS98], or FLORID [LHL$^+$98] support the definition of views consisting of rule sets. Such rule-based views have the advantage that they are easier maintainable and extensible than views defined by single monolithic queries. In contrast to approaches like Strudel, FLORID, HyperView supports the demand-driven integration of data.

Data presentation. Data presentation nowadays primarily means publishing the integrated data in the Web. This amounts to generating HTML or XML documents and serving them to the user's browser on request. A static approach is taken by Strudel which generates all pages of a virtual Web site in advance. In general this is not feasible for external information sources. Moreover, HTML forms cannot be supported this way.

WebOQL uses document patterns to facilitate the formulation of queries that return HTML pages as results. However, this approach aims not so much at defining complex virtual Web sites but at providing certain focused information services.

The PENELOPE language used in the ARANEUS project uses relational nesting operators to format sets of tuples from a relational table into HTML pages.

In YAT, incoming HTTP requests trigger rules that generate HTML pages by matching information in the input data (which may be loaded from external wrappers).

Similarly to YAT, HTTP requests reaching a HyperView virtual Web site trigger rules that produce HTML graphs which are then formatted as HTML documents. Differing from other approaches such as WebOQL and YAT, rules for the presentation layer are specified against the database schema, rather than against the structure of the (HTML) input data.

Completeness. Most of the approaches presented in this chapter do *not* cover all three subtasks necessary for the integration of semistructured information sources, i.e., extraction, integration, and presentation. For instance, TSIMMIS supports data extraction and integration, but not the creation of virtual Web sites. YAT and Strudel do not support data extraction from HTML, even though adding this functionality seems possible. MedMaker, WebOQL, and Strudel are missing a schema concept which makes it difficult to build and maintain complex virtual Web sites. Moreover these approaches are lacking a formal semantics similar to the formal framework of HyperView which forms one of the main contributions of the HyperView approach.

In ARANEUS, heterogeneous approaches are combined into a complete, even though pragmatic solution. This has the disadvantage that different languages and data models are involved. In contrast to this, HyperView offers a homogeneous approach using a graph-based data model and graph-transformation techniques that are applied uniformly throughout all stages. This uniform approach is reflected in the HyperView architecture which consistently uses the same view mechanism to map between different levels of abstraction. This architecture and the HyperView methodology constitute unique contributions of the HyperView approach.

6 Conclusion

A prototypical HyperView for integrating the mentioned wrappers into the conceptual schema depicted in Figure 3 has been implemented. The generic Web HyperView browser can be used to explore the resulting town information database to be explored.

In order to turn this prototype into a professional town information service, the following practical issues have to be dealt with that also apply to other application domains:

Recognizing identical entities: the same event may be announced with different title in several sources. Titles may be abbreviated, in uppercase letters, with additional prefixes etc. Obviously, heuristics are needed for matching records. Work on record linking in databases [Fed97] can be applied in that context. Fusion rules as proposed in [Fau99a] can be used to apply these heuristics and merge matching records.

Classification problems: entities in the database often have to be classified in some way. In our case, cultural projects are classified by genre. The problem is how to achieve this classification. For some focused sources all listed entities can be assigned to the same class due to a-priori knowledge. Other sources provide a classification of their own. Unfortunately, the classes of different sources often do not match.

A pragmatic solution is to mirror all classes occurring in a source in the database. Changes in the classification scheme of a source will reflect immediately in the available classes in the database. However, there may be many synonymous classes from different sources.

Another approach would be to establish a fixed classification hierarchy in the database and mappings from the source-specific classes into this hierarchy. However, changes in the classification scheme of a source will break these mappings.

Moreover, a fixed hierarchy may lead either to an information loss or misclassifications if no exact correspondence between a source-specific class and a class in the database exists. For instance, <tonwerk.de> has a category “Jazz/Funk” which contains both jazz and funk concerts. To add all these events to a category “Jazz” would misclassify the funk concerts. On the other hand, to insert all concerts “Jazz/Funk” to the superclass “Music” would result in information loss. Adding a new class “Jazz/Funk” between “Music” and “Jazz” might result in a proliferation of classes if this is done for all sources.

This is a general data integration problem that goes well beyond the scope of the HyperView approach. Probabilistic methods similar to those proposed for record linkage may help as well in fusing classes with similar names and insert subclass relationships to a number pre-established superclasses. Work on thesauri and ontologies applies as well.

Query optimization: for certain queries, all servers are accessed even though a single server is able to provide all the information. A pragmatic approach is for instance to link genre nodes only to those sources that can provide information for that genre.

Classical query optimization techniques are not sufficient in the context of HyperView due to its semistructured data model and since the incremental view materialization should be taken into account. If a query is subsumed by an earlier query, the result of the new query can be obtained by restricting the materialized result of the earlier query. Hence methods for query optimization with materialized views [CKPS95] and semantic caching [DFJ$^+$96], [KB94] in the relational database setting may be adapted for the semistructured setting.

Navigation-based access: in the current Web interface of the HyperView System, frequently asked types of user queries must be anticipated and implemented by appropriate access paths in the database because the generic HyperView browser offers navigational access but no query interface. For instance, if users want to select all events at a certain date, appropriate computed edges in the database have to be implemented first.

Rapid prototyping and ease of maintenance: currently, schemata and view definitions have to be specified in textual form. However, the underlying graph data-model lends itself easily for visualization and graphical user interfaces. This will be demonstrated by a forthcoming diploma thesis that deals with building a graphical software environment for rapid schema and view specification. Another forthcoming diploma project investigates means for identifying the common structure of a group of pages using data mining methods. This will yield an interactive tool that will assist the view designer in formulating appropriate view definitions.

As we have seen, the HyperView System provides a platform for integrating Web sites in an application area like town information. This platform can be used as a testbed for new heuristics and statistical methods for solving identification and classification problems, and for developing new query optimization techniques and interface concepts.

References

[Abi97] Serge Abiteboul. Querying semi-structured data. In *ICDT'97*, pages 1–18, 1997.

[AM98] G. Arocena and A. Mendelzon. WebOQL: Restructuring documents, databases and webs. In *Proc. of 14th. Intl. Conf. on Data Engineering (ICDE 98)*, 1998.

[Arn99] Arnaud Sahuguet and Fabien Azavant. Wysiwyg Web Wrapper Factory (W4F). Available from the W4F WebSite http://db.cis.upenn.edu/W4F, 1999.

[BDHS96] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu. A query language and optimization techniques for unstructured data. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 505–516, 1996.

[Bun97] Peter Buneman. Semistructured data. In *PODS'97*, 1997. Invited Tutorial.

[CDSS98] Sophie Cluet, Claude Delobel, Jérôme Siméon, and Katarzyna Smaga. Your mediators need data conversion! In *SIGMOD Conference 1998*, pages 177–188, 1998.

[CKPS95] Surajit Chaudhuri, Ravi Krishnamurthy, Spyros Potamianos, and Kyuseak Shim. Optimizing queries with materialized views. In *11th Int. Conference on Data Engineering*, pages 190–200, Los Alamitos, CA, 1995. IEEE Computer Soc. Press.

[DFJ+96] Shaul Dar, Michael J. Franklin, Björn THór Jónsson, Divesh Srivastava, and Michael Tan. Semantic data caching and replacement. In *VLDB'96*, pages 330–341, 1996.

[Fau99a] Lukas C. Faulstich. The formal framework of the HyperView system. Technical Report B 99-03, Inst. of Computer Science, FU Berlin, 1999. <http://www.inf.fu-berlin.de/inst/pubs/tr-b-99-03.abstract.html>.

[Fau99b] Lukas C. Faulstich. *The HyperView Approach to the Integration of Semistructured Data*. PhD thesis, Institute of Computer Science, Free University Berlin, 1999. In preparation.

[Fau99c] Lukas C. Faulstich. Using graph transformation techniques for integrating information from the WWW. In *TAGT'98 final proceedings*, LNCS, 1999.

[Fed97] Federal Committtee on Statistical Methodology. *Record Linkage Techniques – Intl. Workshop*, 1997.

[FFLS97] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language and processor for a web-site management system. In *Workshop on Management of Semistructured Data*, 1997.

[FS99] Lukas C. Faulstich and Myra Spiliopoulou. Building HyperNavigation wrappers for publisher web-sites. *International Journal on Digital Libraries*, 1999.

[KB94] Arthur M. Keller and Julie Basu. A predicate-based caching scheme for client-server database architectures. *The VLDB Journal*, 5(1), 1994.

[KS97] D. Konopnicki and O. Shmueli. W3QS : A system for WWW querying. In *Proceedings of the 13th International Conference on Data Engineering (ICDE'97)*, pages 586–586, Washington - Brussels - Tokyo, April 1997. IEEE.

[LHL+98] B. Ludäscher, R. Himmeröder, G. Lausen, W. May, and C. Schlepphorst. Managing semistructured data with florid: A deductive object-oriented perspective. *Information Systems*, 23(8), 1998. To appear.

[MMM97] A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the World Wide Web. *International Journal on Digital Libraries*, 1(1):54–67, 1997.

[PA97] Paolo Merialdo Paolo Atzeni, Giansalvatore Mecca. To weave the web. In *VLDB '97*, pages 206–215, 1997.

[PGMU96] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. Medmaker: A mediation system based on declarative specifications. In *ICDE'96*, pages 132–141, 1996.

[QWG+96] Dallan Quass, Jennifer Widom, Roy Goldman, Kevin Haas, Qingshan Luo, Jason McHugh, Svetlozar Nestorov, Anand Rajaraman, Hugo Rivero, Serge Abiteboul, Jeffrey D. Ullman, and Janet L. Wiener. LORE: A Lightweight Object REpository for semistructured data. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, page 549, Montreal, Quebec, Canada, 4–6 1996.

[Roz97] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation*. World Scientific Publishing, 1997.

[SL90] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.