A software architecture for generating hypermedia applications for ad-hoc database output

Eindhoven University of Technology
Department of Mathematics and Computing Science

A Software Architecture for Generating Hypermedia Applications for Ad-Hoc Database Output

by

Geert-Jan Houben and Pim Lemmens

99/16

ISSN 0926-4515

editors: prof.dr. J.C.M. Baeten
prof.dr. P.A.J. Hilbers

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Reports 99/16
Eindhoven, November 1999

A Software Architecture for Generating Hypermedia Applications for Ad-Hoc Database Output

Geert-Jan Houben*,** and Pim Lemmens*

* Eindhoven University of Technology
Department of Computing Science
PO Box 513, NL 5600 MB Eindhoven, the Netherlands

** University of Antwerp (UIA)
Department of Mathematics and Computer Science
Universiteitsplein 1, B 2610 Wilrijk, Belgium

E-mail: {g.j.houben,w.j.m.lemmens}@tue.nl

Abstract: This report describes the development of software for the generation of hypermedia (Web) applications for the results of ad-hoc queries to multimedia databases. In [Houben & De Bra 1997] we have proposed a heuristic algorithm for generating navigation structures for multimedia database output, based on ideas from RMM [see Isakowitz et al. 1995]. We have shown that only part of the RMM design methodology can be automated deterministically, and we have provided relevant heuristics and an extension to the query language to allow users to explicitly override the heuristics. In [Houben & De Bra 1999] we have further developed this approach by describing the design of a system for automatic generation of presentations for multimedia database output. In that system hypermedia applications (including both navigation and presentation aspects) are automatically generated combining heuristics from [Houben & De Bra 1997] and explicit directives from the user. This approach is similar to the one used in the DHymE project from [Bieber 1998]. In this report we focus on the architecture of our software, called HERA, that generates hypermedia (Web) applications for legacy data from relational databases.

1. Introduction

In [Houben & De Bra 1997] and [Houben & De Bra 1999] we have addressed an approach to generating World Wide Web applications for volatile information resulting from ad-hoc database queries. The motivation for this approach is that the use of a hypermedia platform such as World Wide Web can help to represent the less structured and not purely textual information one typically finds in applications such as employee databases, museum databases, geographic information systems, and mail-order catalogs and services.

Since the design and construction of a hypermedia application involves the representation of relationships between information objects, the approach for generating the navigation structure is based on the ideas of RMM, specially RMDM [see Isakowitz et al. 1995]. RMM combines elements from the Entity-Relationship model [see Elmasri & Navate 1990] and HDM [see Garzotto et al. 1991] to effectively manage relationships between objects. We are using part of these ideas in software that semi-automatically generates hypermedia presentations, specifically relationships, for data resulting from ad-hoc queries posed to a traditional relational database. To emphasize the volatile character of the data for which relationships need to be generated, we talk about volatile data. Generating a hypermedia presentation for volatile data includes the presentation of the records from the query result and the presentation of a record's relationships with other records. RMM itself and its supporting tool RMCase [see Díaz et al. 1995] do not help in the case of results of arbitrary (ad-hoc) queries. The main problem is that the dynamically generated structure of a query result cannot be (trivially) translated into a hypermedia presentation.

The volatility has led us to a heuristics based routine for the automatic generation of the relationships. In this routine we include information from the data definition part of the database (representing the legacy data structures underlying the volatile data) and from the exact query specification (representing the user's explicit directives). In [Houben & De Bra 1997] and [Houben & De Bra 1999] we have shown how we can deduce the navigation structure for the hypermedia application from the data definition and query specification. In [Houben & De Bra 1998] we have also addressed the presentation aspects that are relevant during the generation process.

While we concentrate in this research on the transformation from relational data to hypermedia presentations, the DHymE project from [Bieber 1998] shows how the (automatic) generation of relationships (e.g. hypermedia links) can be used to add hypermedia functionality to legacy databases. From that point of view our approach can be generalized to add hypermedia functionality to legacy database applications.

The main subject of this report is a description of the architecture for the software implementing our approach. We describe how we handle the communication between a client browser and the underlying database containing the relational data. We also describe how we have set up a software system, called HERA, such that we can generate a hypermedia presentation for the data retrieved from the database. The concrete contents of this report are as follows. First, we start with a background: section 2 discusses general issues involved in the automatic design of hypermedia presentations, in section 3 we address relevant navigation aspects, while section 4 concentrates on the generation process and the heuristics. Then, section 5 contains the main elements of the HERA software architecture.

2. Automatic Hypermedia Presentation Design

There are multiple reasons to produce a mapping from data stored in a relational database to data that can be presented in a hypermedia structure. On the one hand, one can have a legacy (relational)

database full of data for which one wants to produce a presentation that can be accessed through a hypermedia browser (or a Web browser). In that case one wants to produce an alternative presentation for relational data that uses relationships (possibly implemented by hypermedia links) to construct an inner structure for the data that improves the user-friendliness of the presentation. On the other hand it is possible that one has developed a hypermedia application that one wants to populate with data that is available from a legacy database. Either way, one transforms the relational data into data with a hypermedia structure.

Normally, one would want to carefully design a hypermedia presentation by hand. However, when you generate data automatically, then it is not feasible to come up with a presentation designed by hand. This is certainly the case in our applications where we look at results to ad-hoc database queries. Since we cannot foresee exactly which queries are going to be relevant, we cannot come up with a specific hypermedia presentation for each one of them.

We have chosen an approach in which we want to (semi-)automatically derive a hypermedia presentation for these ad-hoc queries. In this approach we use the (heuristic) principle that we want to reuse as much as possible from the result of a manual design process that has generated hypermedia presentations for standard queries on this database. This implies, for example, that we assume that a manual designer has carefully designed a hypermedia presentation to present the contents of each of the base tables. Whenever we then need to construct a hypermedia presentation for an ad-hoc query on a certain base table and it appears that presenting this query result is only a minor variation of presenting that base table, then the generation process would be based on the available presentation for that base table. This is illustrated in figure 1.

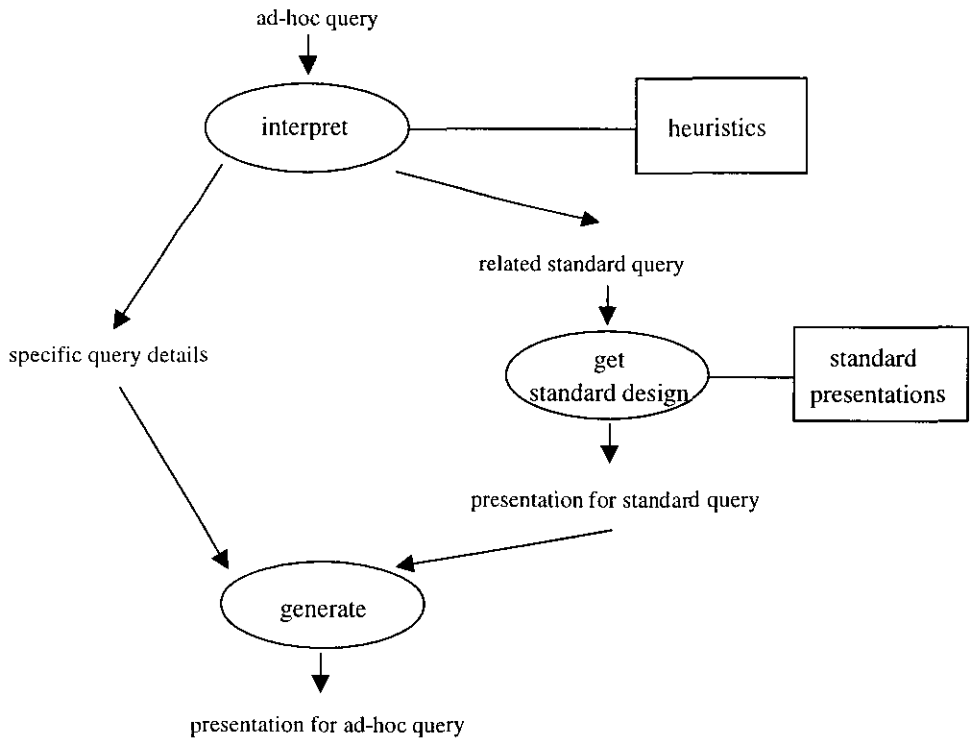

Figure 1: Re-using standard presentations

To facilitate this approach, we use heuristics that represent the "best practice" in generating hypermedia presentations. This means that in the generation process we try to use some "rules" that incorporate knowledge about the way in which query results relate to already available presentations. In this way the heuristics help us to reuse parts of available presentations by mapping ad-hoc queries to standard queries and then subsequently modifying the available presentation for the associated standard query.

3. Navigation

Navigation structures build the most characteristic aspect of a hypermedia application. Compare the approach from DHymE (see [Bieber 1998]) to see the role that navigation structures can play in adding value to a (legacy) application. In the associated method RNA, Relationship-Navigation Analysis, the importance of relationships leads to the use of a rich taxonomy of relationship types in order to assure that the analysis process results in an appropriate set of relationships. From this rich set of relationships RNA produces a set of navigation structures to "implement" the relationships. When a hypermedia platform is used, the navigation structures are realized through hypermedia links.

In our application area we consider data from relational databases. In the generation of a hypermedia application for the legacy database application we add hypermedia links to these data. The nodes in this hypermedia application basically represent the data elements from the database application. The hypermedia links between the nodes represent the structural relationships between those data elements. The motivation for using such a hypermedia representation is the goal to increase the ease with which users can access the information (that was originally stored in the relational database): in addition to the straightforward record presentation other structural relationships between data elements can be made accessible.

For this specific reason we borrow aspects of RMM's data model RMDM [see Isakowitz et al. 1995] in order to express the (structural) relationships between the data elements. RMDM is developed to specify the different mechanisms available for connecting the data elements and thus offering users access to the data elements. The navigation structures that we typically encounter are a combination of hypermedia access routines that relate records to each other, the inter-record navigation, and access routines that relate parts of a single record to each other, the intra-record navigation.

Based on RMDM the mechanisms and tools available for inter-record navigation are: index, guided tour, and indexed guided tour. An index is used to access records by referring to some key identifier. By choosing an identifier from the index the user asks the system to navigate to the corresponding record. Typically the index is composed of words or icons that naturally identify the associated record. A guided tour is used to access records in a given order. The records are connected in a chain-like manner, and the user can follow that chain and thus access the different records. An indexed guided tour is a combination of an index and a guided tour. Essentially, in an indexed guided tour the user has both options to choose from: navigating using the identifiers in the index, or following the predefined path of the guided tour.

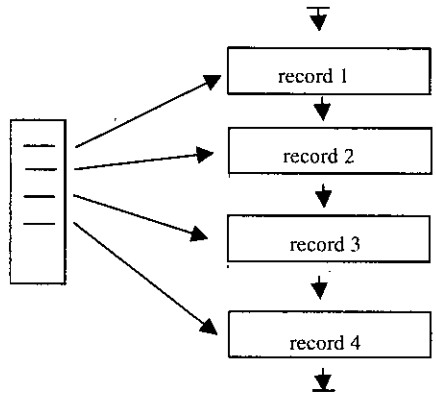

Figure 2: Inter-record navigation

In comparison to RMDM our approach does not include other inter-record navigation mechanisms. This is mainly motivated by the fact that the volatile data collections that result from asking queries are not really special with respect to the inter-record connections: the data collections are always sets of records. The only special detail that seems appropriate is that the user will be able to specify how records are displayed on "pages" in the hypermedia environment. In a hypermedia application we can display one record on one (hyper)page and offer (hyper)links to navigate between such pages. In practice it is often convenient to display a number of records on one page, if the size of the representation of the records allows this. We could use this functionality to offer an additional record-connecting structure: however, we do not address this functionality in this report.

Turning to the intra-record navigation, we use RMDM's "slices" as a way to divide the presentation of a record in multiple parts such that each part can be presented in a natural and easily comprehensible manner. Since the presentation of all properties (attributes) of one record on one hypermedia node (page) often appears not to be feasible because of size restrictions, the designer can divide the record presentation in separate parts. These slices offer a view on a part of a record. In order to move the view towards other parts of the record, there is a link structure available that connects the different slices of a record. These slice links build the intra-record link structure, which is orthogonal to the inter-record connections used to collect sets of records. The definition of this slice link structure is determined at the level of relations: all records of a relation share the same intra-record navigation structure.

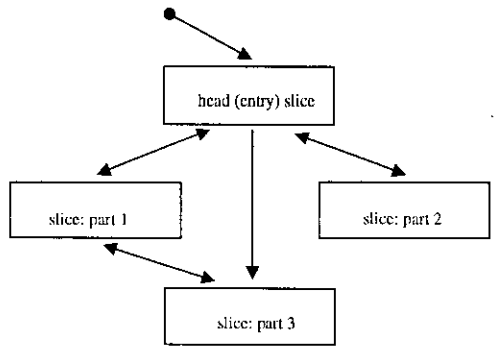

Figure 3: Intra-record navigation

Besides specifying the slices of a record and the links between those slices, we have to determine

how these two kinds of navigation, inter-record navigation and intra-record navigation, are connected. RMDM defines that the slice structure has a head slice, which means that one slice is specified that becomes visible at the time the user "enters" the presentation of the record: the navigation inside the record starts from that head slice.

For more details on our exploitation of RMDM's navigation principles, we refer to [Houben & De Bra 1997].

4. Generation and heuristics

In order to automatically generate a navigation structure for some data we need a fixed routine that the system follows in order to automatically construct this structure. One general heuristic principle underlying our approach implies that the navigation structure to be constructed for the query result depends on (a) the structure embodied in the data from the query result (using some heuristics) and (b) on directives that the user explicitly includes in the query specification itself.

So, the generation process considers first the internal structure contained in the data: the implicit structural relationships within the data serve as a basis for the construction of the navigation structures. This transformation from implicit structural relationships to navigation structures is based on a set of heuristics, that at this moment are fixed in the software. In another part of this research we try to experiment with other sets of heuristics. The current set of heuristics includes rules that for example promote re-use. Acknowledging the fact that human designers have constructed navigational designs for some standard queries, as much as possible of this human effort is re-used when for an ad-hoc query a presentation is generated starting from a standard presentation for a closely associated standard query.

The current heuristics are described in [Houben & De Bra 1999] in detail: that reference includes a number of examples. Basically, the heuristics distinguish between three types of queries:

- Single slice queries are queries in which the user asks for information from a single slice of each of the selected records. There, the heuristics propose to generate a presentation where the inter-record access structure is borrowed from the underlying base table, and where the intra-record access structure is accessed from the single slice that is addressed in the query. The motivation is that the query apparently asks for data from a specific slice.
- Multiple slice queries are queries in which the user asks for information from several slices of each of the selected records. There, the heuristics propose to generate a presentation where again the inter-record access structure is borrowed from the underlying base table, while the intra-record access structure is accessed from the original head slice from that table. Here, the motivation is that apparently the query asks for information from all over the record, therefore the (carefully designed) original intra-record access structure seems most appropriate.
- Multiple relation queries are queries in which the user asks for information from multiple relations within one record, as usually is the case with "joining". The heuristics propose to generate a presentation where the inter-record access structure is borrowed from the first of the underlying base tables, while the intra-record access structure between the different base records is built with an indexed guided tour, and each of these base records is accessed from its head slice. Here, the motivation is that apparently the query retrieves information from different base records: the access structure uses the structures from these original base records to build a "joined" access structure.

By adding explicit directives, the so-called end-user commands, to the specification of the query the user can guide the system in overriding the proposal based on the heuristics. In general, we assume that deviations from the heuristics occur when the user is an advanced user with enough insight into

the application and its underlying database to be able to adjust the representation of the query result. For example, a user can specify that a set of records is accessed through an index and not through the access structure proposed by the heuristics. While eventually the specification of a query will take place in a friendly user interface, for the moment we use an extended version of SQL as the language in which the advanced user is able to specify queries. The extension to SQL facilitates the explicit design directives added by the user to the original query. It does not mean that in the final setting end-users must specify their queries and design commands in this extended SQL. Given the characteristics of the target user group the system will offer the possibility to use a user-friendly Query-By-Example-like user interface that enables the users to specify the necessary details in a convenient manner.

5. Software Architecture

In this section we start the description of the architecture of the software. The software environment, bearing the code name HERA, has two main functions.

1. First, it needs to interpret a user query to find the required data in the database. Next, it needs to present these relational data using hypermedia functionality.
2. In this second step the heuristics and the user's explicit design directives mentioned in the previous sections are exploited to generate a hypermedia presentation: as stated earlier, in this report we concentrate on the navigation aspects in this generation process.

In order to facilitate the design and verification aspects of these two steps, we have thus technically conceived our system as two relatively independent parts, a data manager and a presentation manager.

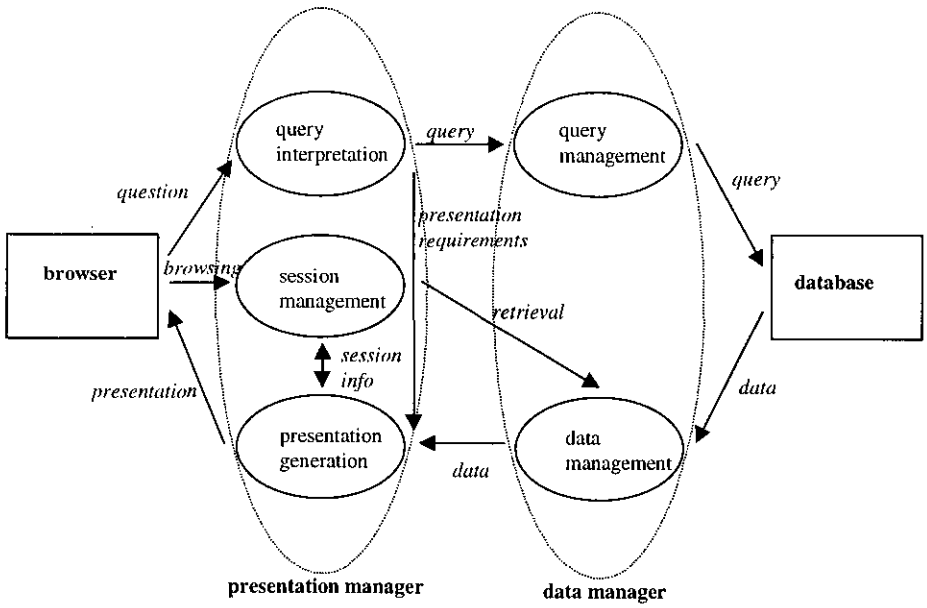

Figure 4: Software architecture

5.1. Presentation Manager

The main task of the presentation manager is to take care of the display of data, hence its name. It presents data to the user, more specifically to the user's browser (in standard HTML pages and forms). In order to do so, it has to generate a hypermedia presentation for the data to be presented. This means that the presentation manager receives data to be displayed from the data manager, it generates a hypermedia presentation for those data on the basis of the heuristics and the explicit design directives,

and it sends this presentation to the user's browser.

Secondly, since the presentation manager communicates with the user's browser, it is also made responsible for the translation of the user input (in the form of HTTP requests) into formal requests for the data manager. The idea here is that the presentation manager handles all the communication between the user and the underlying system, while the user is browsing through the hypermedia presentation that is generated for one query result. This implies that the presentation manager is not only responsible for the processing of the initially asked question, but that during the browsing process it is also responsible for the processing of user actions (e.g. mouse clicks) that lead the user to the presentation of other parts of the same query result. To translate this user input into requests to the data manager the presentation manager should know the state of the user session, what the original query has been and what part of the data the user is viewing.

The generation process involves a number of aspects. The heuristics for the navigation structures have already been discussed, just as the user directives to explicitly override default design suggestions. The actual layout and styling of the presentations is also part of the presentation *manager's task.*

5.2. Data Manager

On the other hand, the data manager translates the formal requests it receives from the presentation manager into queries to the underlying database. So, the data manager communicates with the DBMS of the underlying database and asks this DBMS to retrieve data from its database. This communication involves standard database queries and responses, using JDBC. Subsequently, the data manager delivers the output from the database to the presentation manager.

The output delivered to the presentation manger represents sets of records retrieved from the database. In correspondence with our choice to use RMDM's concept of slice as the basic unit of data, the output is organized in terms of slices. This implies that the output delivered to the presentation manager contains slice data (for the actual data), lists of slice names (for the links to other slices) and index data (for the inter-record navigation). So, as the communication between data manager and presentation manager is realized at the level of slices, actual data values are transferred in terms of slices. The data manager maintains a cache of retrieved data from which it serves the slices the users can ask for.

5.3. Presentation

The data displayed by the presentation manager is arranged in frames and subframes. A frame will typically contain subframes for the actual slice contents, the inter-record navigation and the intra-record navigation.

- The first subframe is the "data" frame. It displays actual data, representing the attribute values from the current slice of the current record. This frame is used for the inspection of the actual data by the user.

- The second subframe is the "inter-record navigation" frame. It presents navigation mechanisms, for example an index showing a list (set) of records, to allow the user to navigate to specific records. These records are the result of the user's query, and together they contain the information the user has asked for. This subframe is meant to facilitate the user in accessing records from that result set. This inter-record navigation subframe can also contain "next" and "previous" buttons to realize a guided tour, if that is the chosen inter-record access structure. As an option (depending on the definitions in the heuristics) an index can be made conditional in the sense that it will not be shown when it contains not enough items.

- The third subframe, the "intra-record navigation" subframe, shows a list (menu) of alternative slices (of the current record). It allows the user to navigate to those slices, thus accessing other parts of the same record. The use of the navigational links in this subframe is basically orthogonal to the use of links in the inter-record navigation frame: this means that the user can navigate through the slices of a record independently from the navigation through the set of records.

- The standard construction of three subframes applies to the queries that ask for records from one single table. In case of a "join" query, where multiple tables are involved, more subframes are made to display all the data slices and navigation slices. Basically, the "joined" record is presented by showing the base records from which the joined record is composed: one base record from each table participating in the join. Therefore, one data slice (subframe) is used per base record to display the data from that record, while for every base record there is one intra-record navigation subframe (to get to its other slices). For the navigation between the participating base records ("within the joined record"), there are subframes that lead to the associated base records. Since the default heuristics propose that an indexed guided tour be used for the association of the records within a join, these subframes will show the links from an indexed guided tour. Besides that, there will be a subframe for the inter-record navigation that offers access to the complete set of joined records: according to the heuristics it does so by offering access to the set of records from the "first" base table, while the other records are accessible through an indexed guided tour.

Within each of these frames there will be a number of fields, each representing some piece of data, that need to be arranged into a meaningful layout and presented in an adequate style. As stated earlier the presentation manager receives the relevant data from the data manager and then uses its own "intelligence" to produce a presentation format. It decides on the actual format for the presentations by considering for example the size of the data items and the number of slices to be presented. The presentation manager can apply certain presentation heuristics to derive a presentation format. In this report we will not elaborate on the presentation heuristics.

5.4. Sessions

The fact that the presentation manager delivers only part of the query result directly to the browser implies that in the technical architecture the presentation manager at first only receives part of the query result (the "first" slice), and that it can subsequently, on the basis of user interaction, ask the data manager for other parts of the query result (other slices). This so-called session management is also part of the responsibility of the presentation manager, since we use the presentation manager for the direct communication with the users. Together with the browsers with which it communicates, the presentation manager will maintain a context for each client. It will however only be concerned with the actual contents of slices and indexes, which is primarily left to the data manager, in so far as it influences their size and style of presentation. Note that since the system is required to concurrently serve an arbitrary number of users, it should keep user records as limited as possible. If it is necessary to store user dependent state information, it will do so in a centrally coordinated manner.

The dialogue mentioned above applies whenever a user is browsing though the presentation generated for a query result. Now, we concentrate on the entire dialogue from the start in more detail.

- When a user starts a session by specifying a question or query, the client will send it as an HTTP Post request to the presentation manager, which wraps it in an XML object that represents the session, and passes it to the data manager.

- The data manager retrieves the query, executes it, puts its response in the session object and

returns it to the presentation manager.

- The presentation manager will pass this session object back to the data manager with each request.

    - When a request involves a choice through an index for an other record, the state of the session object is changed to indicate that the chosen record becomes the "current" record and the data manager returns the record's contents, or rather the contents from the current slice of this record, contained in the updated session object.

    - If the user chooses a different slice of a record, this choice will change the "current slice type" in the session state: this current slice type represents the slice that is considered the current one (for all records). The current record will be unchanged: the "focus" shifts to a different slice, but from the same record.

So, the elements sent from the presentation manager to the data manager are either (initial) queries, or requests for the same slice but from a different record (instance), or requests for a different slice from the same record. In some situations the presentation managers deliver the session object to the data manager to supply the necessary information. In any case, the data manager will in each case respond by returning the (updated) session object.

5.5. Data Transfer

The architecture has been developed in such a way that the data manager does not send the complete set of records contained in the query result: it sends only that portion of the query result that the presentation manager will display, while keeping the rest of the data "in stock". So, the data manager divides the given query result into navigation information, i.e. lists of record keys for the indexes, and in subqueries that contain the data for the individual slices and that may be activated when asked for by the presentation manager. Each time the presentation manager requests a new slice instance, either the same slice of a new record or a new slice of the same record, the data manager will adjust the session data (stored in the session object) to reflect the state change.

In addition to the above-described transformation of user requests into pre-defined slices, the data manager also applies certain heuristics, specially to deal with non-standard user requests. Non-standard user requests are either converted to standard requests (in accordance with the heuristics) or non-standard slices are generated for them. The data manager also contains a caching mechanism that deals with repetitive access to the same data. However, it does not keep track of the state of the individual sessions in which the system is involved.

5.6. Data Types

All in all, there are four types of data involved in the communication between the presentation manager and the database manager: three types of requests from the presentation manager and one type for the session objects that are returned by the data manager and that contain all data relevant to the presentation manager.

1. **Query**: The query is passed from the presentation manager to the data manager in a newly created session object, together with an indication that it concerns a new query. The session object contains an SQL query string with placeholders for parameters whose value depends on choices made by the user. The values resulting from these choices are provided alongside the parameterized query specification.

2. **Session object**: The presentation manager will construct a session object for each separate query. In addition to the query this session object will contain data representing the current state of the session. This includes:
    - data on the record navigation (a frame containing an index)
    - data on the contents (a value frame)
    - data on the slice navigation (a slice menu)
    - additional data in the case of a join query (optional subframes for data from the joined records)

    In addition it indicates which record is the currently chosen record (chosen from the index) and which slice is the currently displayed slice (chosen from the slice menu). For this type of object an XML DTD has been defined. This allows for an easy storage and retrieval on some background medium, as well as an exchange over a network.

3. **Slice request**: An object of this type originates in the presentation manager. It consists of a number that indicates the level of the request (i.e. the position of the base record in a joined record) and the name of the slice type that is chosen at that level.

4. **Record request**: An object of the record request type is passed from the presentation manager to the data manager. It contains an indication of the position of the base record concerned and a number indicating which of the alternatives for that record has been chosen.

5.7. Implementation

A session starts with a request from the user, which is translated to a session object by the data manager. To do this the data manager uses data from the database, which it accesses through JDBC. After the data manager has returned the session object to the presentation manager, it destroys all information on that session, and it will only return to this session after the presentation manager has transferred the session object back for a new request. The presentation manager, upon reception of the session object, starts to build the presentation. It transfers a rendering of all the relevant subframes in the form of HTML frames to the browser. The kinds of frames concerned are: an enveloping frame for each of the base records concerned and within this frame a frame for an index or a set of guided tour buttons, a frame for the actual data of the base record, a menu frame for choosing alternative slices and possibly a subframe for a joined base record. These frames have to be passed to the browser one by one, upon request. Therefore each frame is characterized by a URL which is generated by the presentation manager. The URLs are constructed in such a way that they allow the presentation manager to retrieve the relevant session object and to find the required part of it. The layout of the fields within the frames is implemented using HTML tables or lists. Clickable entities, such as index elements, guided tour buttons or slice menu entries, each receive a URL in which the details of the associated requests are encoded.

After the first set of frames has been constructed the user can interact with the system, clicking on index entries, 'next' or 'previous' buttons, or slice menu entries. The presentation manager receives this response as an HTTP Get request and translates it into requests to the data manager. Every time the user chooses a different alternative for some base record, the information for that record and all successive (joined) base records is updated. A choice for a different slice type at some level results in updating the record information and the slice menu at that level. A user may start a new session by going back to the start page of the presentation and make a new choice of parameters for the query there, whereupon the presentation manager creates a new session object.

For each HTTP request the presentation manager will first decode the URL and then retrieve the session object associated with the session, or create a new one for a new session. When the request has originated from a user interaction, the presentation manager will translate it to a data manager request

and pass it on. Finally it will build a presentation, adding layout and styles to the data, and return it in the form of HTML pages to the browser.

The data manager will, upon receiving a request from the presentation manager, first inspect its cache in search for the required data. When it doesn't find it, it will insert the query parameters into the query string and pass this to the database. The returned data will be cached and the part that was relevant to the request inserted into the session object, which is then returned to the presentation manager.

The entire system is implemented in JAVA, using the JDBC API for database access, the JSDK API to create servlets for the communication between the presentation manager and the browsers, and the SUN Project X XML API for the session object.

6. Conclusion

In this report we have discussed the architecture of a system, called HERA, for the generation of hypermedia applications for data from a relational database. This generation process uses concepts from the RMM paradigm. We have sketched the highlights of our approach and we have concentrated on the software that facilitates this generation process. We have shown that this system's architecture contains two parts, a data manager and a presentation manager, and we have indicated how the functionality of the system is distributed over these parts and how they interact. Specifically, we have shown what data are exchanged and in which order. The result is a flexible and reliable system that allows us to change parts and protocols very easily without compromising the functionality of the other parts. The current software will be very useful for the evaluation of different heuristics, for the generation of both the contents and the presentation of relational data.

7. Acknowledgements

We like to thank Paul de Bra for his contribution to this research and Erik de Kort for his part of the implementation of the preliminary ideas into the first prototype.

8. References

[Borning et al. 1997] Borning, A., Marriott, K., Stuckey, P., Xiao, Y. (1997). Solving linear arithmetic constraints for user interface applications. *Proc. ACM Symposium on User Interface Software and Technology.*

[Bieber 1998] Bieber, M., Hypertext and Web Engineering, *Proc. ACM Hypertext'98*, ACM Press, 277-278.

[Díaz et al. 1995] Díaz, A., Isakowitz, T., Maiorana, V., Gilabert, G. (1995). RMC: A Tool to Design WWW Applications. *Proc. Fourth International World Wide Web Conference,* 559-566.

[Elmasri & Navate 1990] Elmasri, R. & Navate, S. (1990). *Fundamentals of Database Systems.* The Benjamin/Cummings Publishing Company, second edition.

[Garzotto et al. 1991] Garzotto, F., Paolini, P., Schwabe, D. (1991). HDM - A Model for the Design of Hypertext applications. *Proc. ACM Hypertext '91 Conference*, ACM Press, 313-328.

[Houben & De Bra 1997] Houben, G.J. & De Bra, P. (1997). World Wide Web Presentations for Volatile Hypermedia Database Output. *Proc. WebNet97, the World Conference of the WWW, Internet,and Intranet,* AACE, 229-234.

[Houben & De Bra 1999] Houben, G.J. & De Bra, P. (1999). Retrieval of Volatile Database Output through Hypermedia Applications. *Proc. 32nd Hawaii International Conference on Systems Sciences*, IEEE Computer Society, CD-ROM.

[Isakowitz et al. 1995] Isakowitz, T., Stohr, E., Balasubramanian, P. (1995). RMM: A Methodology for Structured Hypermedia Design. *Communications of the ACM*, 38 (8), 34-44.

Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

96/01	M. Voorhoeve and T. Basten	Process Algebra with Autonomous Actions, p. 12.
96/02	P. de Bra and A. Aerts	Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12
96/03	W.M.P. van der Aalst	Parallel Computation of Reachable Dead States in a Free-choice Petri Net, p. 26.
96/05	T. Basten and W.M.P. v.d. Aalst	A Process-Algebraic Approach to Life-Cycle Inheritance Inheritance = Encapsulation + Abstraction, p. 15.
96/06	W.M.P. van der Aalst and T. Basten	Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.
96/07	M. Voorhoeve	Structural Petri Net Equivalence, p. 16.
96/08	A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk	OODB Support for WWW Applications: Disclosing the internal structure of Hyperdocuments, p. 14.
96/09	F. Dignum, H. Weigand, E. Verharen	A Formal Specification of Deadlines using Dynamic Deontic Logic, p. 18.
96/10	R. Bloo, H. Geuvers	Explicit Substitution: on the Edge of Strong Normalisation, p. 13.
96/11	T. Laan	AUTOMATH and Pure Type Systems, p. 30.
96/12	F. Kamareddine and T. Laan	A Correspondence between Nuprl and the Ramified Theory of Types, p. 12.
96/13	T. Borghuis	Priorean Tense Logics in Modal Pure Type Systems, p. 61
96/14	S.H.J. Bos and M.A. Reniers	The I^2C-bus in Discrete-Time Process Algebra, p. 25.
96/15	M.A. Reniers and J.J. Vereijken	Completeness in Discrete-Time Process Algebra, p. 139.
96/17	E. Boiten and P. Hoogendijk	Nested collections and polytypism, p. 11.
96/18	P.D.V. van der Stok	Real-Time Distributed Concurrency Control Algorithms with mixed time constraints, p. 71.
96/19	M.A. Reniers	Static Semantics of Message Sequence Charts, p. 71
96/20	L. Feijs	Algebraic Specification and Simulation of Lazy Functional Programs in a concurrent Environment, p. 27.
96/21	L. Bijlsma and R. Nederpelt	Predicate calculus: concepts and misconceptions, p. 26.
96/22	M.C.A. van de Graaf and G.J. Houben	Designing Effective Workflow Management Processes, p. 22.
96/23	W.M.P. van der Aalst	Structural Characterizations of sound workflow nets, p. 22.
96/24	M. Voorhoeve and W. van der Aalst	Conservative Adaption of Workflow, p.22
96/25	M. Vaccari and R.C. Backhouse	Deriving a systolic regular language recognizer, p. 28
97/02	J. Hooman and O. v. Roosmalen	A Programming-Language Extension for Distributed Real-Time Systems, p. 50.
97/03	J. Blanco and A. v. Deursen	Basic Conditional Process Algebra, p. 20.
97/04	J.C.M. Baeten and J.A. Bergstra	Discrete Time Process Algebra: Absolute Time, Relative Time and Parametric Time, p. 26.
97/05	J.C.M. Baeten and J.J. Vereijken	Discrete-Time Process Algebra with Empty Process, p. 51.
97/06	M. Franssen	Tools for the Construction of Correct Programs: an Overview, p. 33.
97/07	J.C.M. Baeten and J.A. Bergstra	Bounded Stacks, Bags and Queues, p. 15.
97/08	P. Hoogendijk and R.C. Backhouse	When do datatypes commute? p. 35.

97/09	Proceedings of the Second International Workshop on Communication Modeling, Veldhoven, The Netherlands, 9-10 June, 1997.	Communication Modeling- The Language/Action Perspective, p. 147.
97/10	P.C.N. v. Gorp, E.J. Luit, D.K. Hammer E.H.L. Aarts	Distributed real-time systems: a survey of applications and a general design model, p. 31.
97/11	A. Engels, S. Mauw and M.A. Reniers	A Hierarchy of Communication Models for Message Sequence Charts, p. 30.
97/12	D. Hauschildt, E. Verbeek and W. van der Aalst	WOFLAN: A Petri-net-based Workflow Analyzer, p. 30.
97/13	W.M.P. van der Aalst	Exploring the Process Dimension of Workflow Management, p. 56.
97/14	J.F. Groote, F. Monin and J. Springintveld	A computer checked algebraic verification of a distributed summation algorithm, p. 28
97/15	M. Franssen	λP-: A Pure Type System for First Order Loginc with Automated Theorem Proving, p.35.
97/16	W.M.P. van der Aalst	On the verification of Inter-organizational workflows, p. 23
97/17	M. Vaccari and R.C. Backhouse	Calculating a Round-Robin Scheduler, p. 23.
97/18	Werkgemeenschap Informatiewetenschap redactie: P.M.E. De Bra	Informatiewetenschap 1997 Wetenschappelijke bijdragen aan de Vijfde Interdisciplinaire Conferentie Informatiewetenschap, p. 60.
98/01	W. Van der Aalst	Formalization and Verification of Event-driven Process Chains, p. 26.
98/02	M. Voorhoeve	State / Event Net Equivalence, p. 25
98/03	J.C.M. Baeten and J.A. Bergstra	Deadlock Behaviour in Split and ST Bisimulation Semantics, p. 15.
98/04	R.C. Backhouse	Pair Algebras and Galois Connections, p. 14
98/05	D. Dams	Flat Fragments of CTL and CTL*: Separating the Expressive and Distinguishing Powers. P. 22.
98/06	G. v.d. Bergen, A. Kaldewaij V.J. Dielissen	Maintenance of the Union of Intervals on a Line Revisited, p. 10.
98/07	Proceedings of the workhop on Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98) June 22, 1998 Lisbon, Portugal	edited by W. v.d. Aalst, p. 209
98/08	Informal proceedings of the Workshop on User Interfaces for Theorem Provers. Eindhoven University of Technology ,13-15 July 1998	edited by R.C. Backhouse, p. 180
98/09	K.M. van Hee and H.A. Reijers	An analytical method for assessing business processes, p. 29.
98/10	T. Basten and J. Hooman	Process Algebra in PVS
98/11	J. Zwanenburg	The Proof-assistemt Yarrow, p. 15
98/12	Ninth ACM Conference on Hypertext and Hypermedia Hypertext '98 Pittsburgh, USA, June 20-24, 1998 Proceedings of the second workshop on Adaptive Hypertext and Hypermedia.	Edited by P. Brusilovsky and P. De Bra, p. 95.
98/13	J.F. Groote, F. Monin and J. v.d. Pol	Checking verifications of protocols and distributed systems by computer. Extended version of a tutorial at CONCUR'98, p. 27.
98/14	T. Verhoeff (artikel volgt)	
99/01	V. Bos and J.J.T. Kleijn	Structured Operational Semantics of χ , p. 27
99/02	H.M.W. Verbeek, T. Basten and W.M.P. van der Aalst	Diagnosing Workflow Processes using Woflan, p. 44
99/03	R.C. Backhouse and P. Hoogendijk	Final Dialgebras: From Categories to Allegories, p. 26
99/04	S. Andova	Process Algebra with Interleaving Probabilistic Parallel Composition, p. 81

99/05	M. Franssen, R.C. Veltkamp and W. Wesselink	Efficient Evaluation of Triangular B-splines, p. 13
99/06	T. Basten and W. v.d. Aalst	Inheritance of Workflows: An Approach to tackling problems related to change, p. 66
99/07	P. Brusilovsky and P. De Bra	Second Workshop on Adaptive Systems and User Modeling on the World Wide Web, p. 119.
99/08	D. Bosnacki, S. Mauw, and T. Willemse	Proceedings of the first international syposium on Visual Formal Methods - VFM'99
99/09	J. v.d. Pol, J. Hooman and E. de Jong	Requirements Specification and Analysis of Command and Control Systems
99/10	T.A.C. Willemse	The Analysis of a Conveyor Belt System, a case study in Hybrid Systems and timed μCRL, p. 44.
99/11	J.C.M. Baeten and C.A. Middelburg	Process Algebra with Timing: Real Time and Discrete Time, p. 50.
99/12	S. Andova	Process Algebra with Probabilistic Choice, p. 38.
99/13	K.M. van Hee, R.A. van der Toorn, J. van der Woude and P.A.C. Verkoulen	A Framework for Component Based Software Architectures, p. 19
99/14	A. Engels and S. Mauw	Why men (and octopuses) cannot juggle a four ball cascade, p. 10
99/15	J.F. Groote, W.H. Hesselink, S. Mauw, R. Vermeulen	An algorithm for the asynchronous *Write-All* problem based on process collision*, p. 11.