World Wide Web Presentations for Volatile Hypermedia Database Output

Geert-Jan Houben[1] and Paul De Bra[2]
Department of Computing Science
Eindhoven University of Technology
The Netherlands
{houben,debra}@win.tue.nl

Abstract: The use of databases in current information systems is changing rapidly. The information contained in the database shows less structure, and contains multi-media objects and free-text, as well as hypertext links. To represent the information from a database a hypermedia platform such as World Wide Web often appears to be a right choice. Typical examples of such hypermedia applications are employee databases that include both administrative and personal information, museum databases that offer guided tours as well as query facilities for their collection, geographic information systems as used in tourist applications, and mail-order catalogs and services.

The Relationship Management Methodology (RMM) [Isakowitz et al. 1995] is a hypermedia design method for generating hypermedia navigation and presentations of database information. However, most of the output of databases is the result of queries. Such output does not have a predefined structure for which a representation could be designed using RMM. In this paper we present an approach for extending the principles of RMM in order to generate hypermedia (HTML) presentations and navigational structures for query answers. First, we propose heuristics that are based on Relationship Management Design Model (RMDM) structures generated through RMM to provide presentational and navigational cues in a database query. Moreover, we give ways to override these defaults through extensions to the SQL query language.

1. Introduction

The way in which databases are used in current information systems often differs significantly from the way we are used to. The information contained in the database is less strictly structured than in traditional (administrative) information systems. The use of a hypermedia platform such as World Wide Web can help to represent the less structured information. Typical examples of such applications are employee databases, museum databases, geographic information systems, and mail-order catalogs and services. The design and construction of such hypermedia applications for database information is the main subject of this paper.

Designing and constructing a hypermedia application involves the representation of relationships among information objects. The Relationship Management Methodology (RMM) [Isakowitz et al. 1995] is based on the Entity-Relationship model [Elmasri et al. 1990] and on HDM [Garzotto et al. 1991]. It proposes a methodology to support the design and construction of an application by suggesting a translation from an E-R design to a navigational design with RMDM (RMM's data model) constructs. The methodology is augmented with RMCase [Díaz et al. 1995], a CASE tool that provides not only a graphical interface for developing RMDM constructs, but also an interface for designing the presentation of an RMDM construct as an HTML document. Note that RMM is intended for multimedia databases that combine free (or almost free) text, images and possibly also other information. For "administrative" data the generation of a hypermedia (really hypertext) representation is much easier, as demonstrated by [Pönighaus et al. 1996].

RMM includes a translation from E-R models to RMDM models. In RMDM E-R relationships are replaced by navigational structures such as indexes and guided tours. Entities, especially large entities, are divided into slices for better presentation. Slices are groups of attributes which belong together semantically and which are

[1]Geert-Jan Houben is also affiliated with Origin.
[2]Paul De Bra is also affiliated to the University of Antwerp, and with the "Centrum voor Wiskunde en Informatica" in Amsterdam.

thus presented simultaneously. For each entity there is a head slice, presenting the most essential information, and containing (hypertext) links to the other slices.

The subject of our interest is the presentation (through WWW browsers) of volatile database output. This volatile information is the result of queries executed by the database. Query results depend on two aspects:

- the definition of the data in the database, i. e. the (static) data structures that underly the volatile information;
- the specification of the exact query, i.e. the dynamically determined aspects that are necessarily attached to the volatile information produced by the specific query.

We find that RMM (and RMCase) do not help in the case of volatile data. The main problem is that the dynamically generated structure of a query result cannot be (trivially) translated into a hypermedia presentation.

- The RMM approach can be applied to applications of a fairly stable nature. For the hypermedia applications based on more volatile information the RMM guidelines fall short since much of the information about the database, and especially the dynamically determined aspects of the query results, is not explicitly contained in the associated data model. So, RMM can help to guide a designer to find a proper representation by using the data definition. However, the exact specification of the query needs to be considered as well to produce a fully suitable hypermedia representation.
- Also, since the number of possible data structures of query results is very large it is not feasible to create the navigational structure and the HTML presentation through RMCase for each possible query. In [Díaz et al. 1995] the designers of RMCase acknowledge that the creation of hypermedia representations for database queries is a complex task, not handled by RMCase.

The core of our proposal in this paper is to use as much as possible the presentation aspects that are related to the data definition. However, in addition to the use of the data structures this paper includes general guidelines to present the dynamic, query dependent information in hypermedia format. In order to be able to do so, we must consider the information from the database in both the data definition and the data manipulation perspective. We will use SQL as the language to specify data definition and manipulation. We give a translation from SQL to RMDM while using the existing translation from E-R to RMDM. Thus, we present an extension to the RMM methodology that bridges the gap between SQL and RMDM. The extension to RMM is twofold:

1. We show how the data definition component of the database can be used to propose a default representation in RMDM structures for any query result.
2. We suggest an extension to SQL (to be interpreted by a preprocessor) to offer the user the possibility to include representation information (i.e. navigation and presentation) in a query.

2. RMM and SQL

As we have seen in the previous section, RMM offers a translation from E-R to RMDM. This translation is elegantly presented in [Isakowitz et al. 1995]. For the purpose of representing the structural database information by means of a hypermedia format, this translation (we call it ERRep) suffices. To be able to translate queries (dynamically determined information) we must take the data manipulation aspects into account as well. As we choose to use SQL as the vehicle for data definition and manipulation, SQL is the source platform for the translation to RMDM.

We need to consider another translation, the one from SQL to E-R. In practice an E-R diagram needs to be translated to a set of relations (in the relational database model) in order to obtain an implementation. This translation from the E-R model to the relational model is straightforward (and often done according to a standard method). The inverse translation is equally simple: it is this translation (called RMRep) from the relational model to the E-R model that we must consider to complete the global translation from SQL (the relational model) to RMDM. We assume here that we know how entities and relationships are modeled by the given database relations.

These two translations ERRep and RMRep build the mechanism to translate the major data definition elements into RMDM representations. For each relation in the database, specified in the SQL data definition part, a representation in RMDM is generated by applying the translations ERRep and RMRep.

For the sake of translating data definition aspects, this approach suffices. We can not apply this approach to the data manipulation aspects. When we consider the volatile information specified by SQL queries, the translation through E-R can not be used as an intermediate step in all circumstances. In general, this E-R intermediate is not available for data manipulation: we then must be able to use a direct translation from the SQL context to RMDM.

3. Hypermedia Representation and Data Definition

To obtain a generally applicable translation mechanism, we have found it effective to use as much information as possible from the data dictionary to get the core of the representation in hypermedia format (RMDM). Thus, the representation that we propose for volatile database output is based on a default representation determined by the data definition aspects. Subsequently, the default (standard) representation is adjusted to the specific details in the data manipulation aspects, i. e. the dynamically determined aspects of the information.

To be able to deduce an elegant hypermedia presentation, details should be included in the data definition that concern the user preferences for the representation aspects. A number of issues concern the relations involved:

- The user can specify for every relation how the set of records (instances) is presented: one per page, all on the same page, or a fixed number of records presented on one page. By determining how many records are shown on one page, the user has the possibility to adjust the presentation to the semantics and the size of the records.
- Related to the issue of the presentation of records on pages, is the issue of the connection between those pages: the user must specify the access structure of the relation, and thus the way in which the different pages are accessed. By specifying in the data definition the standard access structure for each relation, like index or guided tour, the user can express the access routine that best suits the way in which users want to access the different pages.
- The above two issues must be determined for the presentation of the relation itself, but also for query results that are a subset of the entire relation. As the presentation of a query result might differ from the general presentation, we give the user the possibility to add additional information related to the size of the query result. The user can specify a different number of records per page and a different access structure, if the number of records to show is less than a given number. The idea is that it could be wise to present a small number of records on one page, while presenting large numbers of records through an index.

RMM associates a set of slices with every relation. Some presentation issues concern the slices:

- In order to be able to access the different slices, an access structure for the slices of a relation must be specified. It must be determined how slices are accessed from the head slice. In this way the different semantics attached to the different slices can be expressed.
- Besides the access structure for the slices, the user must specify for every slice which presentation (layout) should be used. This implies that the user determines how a specific slice is presented in hypermedia format.
- In the same way as for relations the presentation can depend on the size of the (computed) relation; hence we allow the user to specify how the slice presentation and access structure depends on the number of slices (records) to show. This includes the number of slices (records) presented on one page and the access structure for those pages.

As we will see later, the presentation of database output (query results) implies the creation of new slices that contain (only) the information explicitly asked for in a query. For these newly created (volatile) slices presentation aspects can be defined in the data definition:

- The user should have the possibility to specify beforehand the default presentation framework (layout) to be used for new automatically generated slices. This framework can later be used to determine the actual presentation of a specific query result. Of course, this framework can be defined in general (for all relations), but we allow the user to override it by specifying a specific framework for a given relation.
- As before for the relations and (structural) slices, for the other presentation issues (access structure and records per page) a default can be proposed for representation of the volatile slices. Again, this default can be overridden in the case of specific query, but one can expect that per relation a default can be determined beforehand for its volatile slices.

The nature of this approach is that during the data definition a choice is made between for example index and guided tour, based on the semantics and syntax (layout) of the items (entities or relationships) involved. Moreover, based on the same semantics and syntax the most elegant combination of records on separate pages is chosen. The general idea in this representation is that most presentation issues are settled beforehand, thus giving a default (framework) that can act as the basis for determining the exact presentation during data manipulation.

4. Hypermedia Representation and Data Manipulation

In this section we concentrate on the routine that is used to produce a hypermedia representation during data manipulation. The prime idea behind this routine is that the user is offered a predefined, default representation, based on the presentation aspects included in the data definition. The input for the routine during data manipulation is an SQL query. Our routine includes three main steps:

1. The exact query specification is considered to see what the result of the query can borrow from the RMDM representation issues of the relations closely related to that result: these issues are available in the data (definition) dictionary and act as the default to start from.
2. The default representation deduced from the data definition is adjusted in correspondence with the number of records found in the result.
3. The proposed representation is overridden and adjusted, if the query specification contains elements that explicitly ask for a given representation.

In this section we now address the first two steps: the third step is covered in the next section. To illustrate the use of predefined representation aspects we must start the data manipulation from a given SQL query. Let us assume that this query looks like

```
SELECT <Attributes>
FROM <Relations>
WHERE <Condition>
```

The outline of the routine distinguishes three cases.

4.1 One Slice, One Relation

In this first case the attributes in <Attributes> all belong to one slice, say S, of a relation, say R. If the user specifically asks for attributes from one slice, this slice is offered (directly accessible) to the user.

As far as the access structure for these records (slices) is concerned, we choose to borrow the representation of R (from data definition): so, we use slice S as the entry slice to the relevant records, which are accessible to their default access structure (from data definition). At the same time, the head slices remain also connected through that similar access structure.

We assume here that in general every slice is designed in such a way that the user is satisfied with the presentation of the entire slice (and does not feel the need to limit the presented data). So, the fact that the user asks (in the SELECT clause) for some of the attributes from a given slice, is interpreted as asking for that (whole) slice.

4.2 Multiple Slices, One Relation

Another possibility is that the attributes in <Attributes> belong to multiple slices of one relation R and <Relations> only contains R. We then use the idea that if the user asks for attributes from multiple slices, the user should be offered the relevant records through their head slices, with additional access to a new (volatile) slice with exactly the attributes from <Attributes>. The new slice gets a default name, e.g. NEWSLICE. So, the representation looks like the representation of R for the relevant records, with the head slice as its entry slice and one additional (volatile) slice containing all the attributes of <Attributes>.

As far as the access structure between records is concerned, we choose to use the same principle as above. This means that the default access structure between records is borrowed from the data definition, and it is also implemented to connect the new volatile slices. Just as in the previous case, we obtain two record connecting access structures: one between the head slices, and one specially for the slices implied in the query.

To the access structure between slices (within a record) we add two links: one from the head slice to the new volatile slice, and one link in the opposite direction. The default names are HEAD-NEW and NEW-HEAD.

Just as in the previous case, if the user does not want to see a new slice, the user simply accesses the relevant records through their standard head slices: apparently the user accepts that the access structure between records and slices will guide the user to the relevant data. It is only when the user explicitly wants the selected attributes (and nothing else) that the user must refer to the new automatically generated slice. However, we acknowledge this possibility that the user wants to primarily access the volatile slices: therefore, we offer the additional record access structure between the volatile slices.

4.3 Multiple Relations

The third general case is that the attributes in <Attributes> belong to multiple relations. We feel that such a query expresses a relationship between records (relations). So, besides determining how records and slices are presented, we must determine how this relationship between records is presented. We propose that if there are more relations involved, each relation can be accessed through an indexed guided tour from the previous relation (in terms of the sequence in <Relations>). This means if <Relations> equals R1, R2, that from a record from R1 the associated records from R2 are accessible through an indexed guided tour.

As default representation for records and slices, we borrow their standard representation. Of course, we also want to express the information explicitly composed in the SELECT clause. We do so by adding to the records of the first relation R1 a new volatile slice with only the attributes in <Attributes>. This new slice obtains a default name, e.g. NEWSLICE, and it is connected through two links to the head slice of R1, e.g. HEAD-NEW and NEW-HEAD. These new volatile slices are connected using the default record access structure from R1.

The principle underlying this part of the routine is that a user-specified association between records can best be represented by accessing the original relations via indexed guided tours. By offering a new automatically generated slice (page) with the explicitly selected attributes, the user can choose between this limited selection of information and a standard access structure offering a overview of the selected record associations.

5. Adding Representation Information to Queries

The third step of our routine allows the user to explicitly override and adjust the default presentation (deduced in the first two steps) through "hints" in the query. While the first two steps use the principles of RMM, together with heuristics on the translation from SQL to RMDM, in the third step we allow the user to disagree with the proposed interpretation of the query semantics. It is important to note that in practice the user who composes queries may be different from the user (designer) responsible for the data definition.

Our third step assumes that we can use a preprocessor of the SQL queries that can interpret its input and produce (an SQL query and) user-specific presentation information. The purpose of this preprocessor is to combine the default representation from the data definition with the query dependent details to produce the presentation which the user likes best.

If we look at the first part of the routine (Section 4.1) we see that the user gets the slice from which he has selected some attributes. So, while the user may have specified a smaller number of attributes, he is presented with the entire slice. If the user explicitly wants this limited view of the slice, he should be able to specify that. For this reason we allow the user to write

```
SELECT NEW <Attributes>
```

in the query. The preprocessor interprets the NEW command in such a way that a new volatile slice is created with exactly the specified attributes. This new slice is accessible from the head slice, but it is presented as the entry slice.

The new volatile slice gets a default name, e. g. NEWSLICE. If the user explicitly wants to override that name, he can add the construct NAME <name> to the SELECT clause. Writing

```
SELECT <Attributes> NAME <name>
```

causes that <name> becomes the new name for the volatile slice.

Similarly, the user can override the default names for the links to and from the new slice. By adding the construct FORWARD LINK <name> and BACKWARD LINK <name>, the user can specify the new names for these links:

```
SELECT <Attributes> NAME <name> FORWARD LINK <link-name>
```

The last two parts of the routine (Section 4.2-3) show that in most cases a volatile slice is created. In a number of situations the user may not be interested in this slice, but is only interested in the selected associations between records. In that case the user may help the database system by notifying it that a new slice need not be created. By writing

```
SELECT NO NEW <Attributes>
```

in the query, the preprocessor is instructed to omit the creation of a new slice.

All three parts of the routine (Section 4.1-3) show that in general the default presentation from the data definition is used. The user may want to override that information in the query formulation.

Writing INDEX or TOUR after a relation name in the FROM clause causes the database system to use an index or guided tour as the access structure for that relation. The next example causes the Person records to be accessible through an index:

```
FROM Person INDEX
```

By writing INDEX or TOUR before a relation name (not the first) in the FROM clause, the database system will use the index or guided tour as the way to present the association between the relations. If for example the FROM clause looks like

```
FROM Person, INDEX Employee
```

then the association between Person and Employee records is given through an index (instead of the default indexed guided tour).

The use of PAGE(x) after a relation name in the FROM clause, can override the default for the number of slices (records) shown on one page. In this case x is an integer (minimum 1), while we allow the user to write PAGE(*) to denote that all slices are shown on one page. The next expression results in one Hobby slice per page, if the attributes in the SELECT clause specify the Hobby slice:

```
FROM Person PAGE(1)
```

Another aspect that can be influenced by the user is the implementation of the new record connecting access structure between the volatile slices. If the user is satisfied with the record access structure that only connects the head slices, the user can specify the construct HEAD ONLY in the FROM clause:

```
FROM HEAD ONLY <Relations>
```

Then only the head slices get connected (through the default access structure).

6. Conclusion and Future Work

In this paper we have described the outline of a routine to produce a hypermedia representation for volatile database output, specified by an SQL query. The routine uses the default representation specified in the data dictionary, but the user can override and adjust this default. It is important to realize that different people are involved. The designer of the system has the task to produce a suitable design of the query results that corresponds both to the semantics of the data queried and to the semantics of the query result itself. The designer must acknowledge that the person posing a query can override that design in order to efficiently use the query result in his work.

We are currently involved in a number of projects that require hypermedia (World Wide Web) presentation and navigation for information systems applications. One project concentrates on query interfaces for geographic information systems. A second project aims at providing travel, lodging and entertainment information for the disabled tourist in Europe. A third project aims at disclosing all museum information in the Netherlands through a Web server, including a general-purpose query interface for art- and history researchers. We intend to evaluate the proposed routine for generating hypermedia representations of database query results during the development phase of these projects.

The CD-ROM version of this paper contains some more illustrations to clarify our approach. These had to be eliminated in the paper version for lack of space.

References

[Díaz et al. 1995] Díaz, A., Isakowitz, T., Maiorana, V. and Gilabert, G., RMC: A Tool to Design WWW Applications. Proceedings of the Fourth International World Wide Web Conference, pp. 559-566, Boston, 1995.

[Elmasri et al. 1990] Elmasri, R. and Navate, S., Fundamentals of Database Systems. The Benjamin/Cummings Publishing Company, second edition, 1990.

[Garzotto et al. 1991] Garzotto, F., Paolini, P. and Shwabe, D., HDM - A Model for the Design of Hypertext applications. Proceedings of the ACM Hypertext'91 Conference, ACM Press, pp. 313-328, 1991.

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

[Pönighaus et al. 1996] Pönighaus, R. and Mitlöhner, J., Relational Databases and the World Wide Web: Automatic Generation of Hypertext based on Reverse-engineered Meta Information. Proceedings of WebNet'96, the World Conference of the Web Society, pp. 401-406, San Francisco, 1996.