Journal of Universal Computer Science, vol. 1, no. 10 (1995), 687-705
submitted: 11/7/95, accepted: 14/11/95, appeared: 28/10/95©Springer Pub. Co.

Contained Hypermedia

Erik Duval
(Departement Computerwetenschappen, Katholieke Universiteit Leuven, Belgium
Erik.Duval@cs.kuleuven.ac.be)

Henk Olivié
(Departement Computerwetenschappen, Katholieke Universiteit Leuven, Belgium
olivie@cs.kuleuven.ac.be)

Nick Scherbakov,
(Institute for Information Processing and Computer Supported New Media (IICM),
Graz University of Technology, Austria
nsherbak@iicm.tu-graz.ac.at)

Abstract: We propose a new hypermedia data model, called CHM for Contained HyperMedia. Our model is based on set-oriented data structuring, with a strong emphasis on automatic maintenance of link integrity. In this paper, the CHM model is presented in detail: both data structuring, navigational facilities and authoring support are presented. We will also explain how we have integrated support for the CHM model in Home, our Hypermedia Object Management Environment, publicly accessible through the World-Wide Web.

Key Words: hypermedia data modeling, automatic link maintenance
Category: H.2.1,H.5.1, I.7.2

1 Introduction

The most simple hypermedia data model is the *basic node-link paradigm*: information is organized in chunks, usually called 'nodes' and interrelated by 'links' [Conklin 1987, Nielsen 1990]. As has been reported before [Andrews, et al. 1995b, Duval, Olivié 1995, Maurer, et al. 1994, Srinivasan, Scherbakov 1995], the simplicity of the basic node-link paradigm leads to a number of problems:

- navigating the links between nodes, users quickly become disoriented (the so-called *"lost in hyperspace"* problem);
- manual *link maintenance* is a tedious burden in large-scale hypermedia systems: whenever e.g. a node is deleted, all references to it must be updated or dangling references will arise;
- most hypermedia systems based on the basic node-link paradigm (notably the World-Wide Web [Berners-Lee, et al. 1994, Cailliau 1995], hereafter referred to as WWW) emphasize *retrieval rather than modification* of information and *individual rather than cooperative* creation and use of the information.

The basic challenge in any attempt to overcome these problems is to propose additional or alternative data structuring facilities, while retaining as much as possible the simplicity and flexibility of the basic node-link paradigm.

For this purpose, we have developed a new hypermedia data model, based on set-oriented data structuring, with a strong emphasis on automatic maintenance of link integrity. Our model is strongly influenced by early work in this area [Parunak 1991], the HM data model [Andrews, et al. 1995c, Maurer, et al. 1994, Srinivasan, Scherbakov 1995] and the SOS model [Duval, et al. 1995].

Our hypermedia model is supported in Home, our Hypermedia Object Management Environment [Duval, et al. 1995, Duval, Olivié 1995]. Home can be considered as a second-generation WWW server [Andrews, et al. 1995b]. In fact, we rely on an ordinary WWW server and use the Common Gateway Interface (CGI) mechanism to add functionality to the server. Home supports:

- more sophisticated data structuring and navigation facilities, so that a *more structured view on the server content* is presented to end users;
- *automatic support of link integrity*, using a set based hypermedia model with automatically maintained navigational topologies, as will be explained further on;
- navigational and query based *access* to the server contents, as well as *authoring* facilities, all through ordinary WWW clients.

The remainder of this text is structured as follows: section 2 introduces the CHM data model. Section 3 deals with navigational facilities in the CHM model. The authoring process is described in section 4. In section 5, we present the concept of "slots". Some implementation aspects are dealt with in section 6. Section 7 compares the results described here with some of our previous work and with related work in general. Before the final conclusion, we mention some of our plans for the future in section 8.

2 Contained Hypermedia Model

2.1 Informal Overview

In the CHM model, content is encapsulated in *multimedia objects*. These are not specified in detail; their only property relevant here is that they can be 'visualized'. This may for instance correspond to showing an image, a video clip or a computer animation. Visualisation of multimedia objects should be understood in a very loose sense here: it can refer to a non-visual action like playing a sound sequence or a more complex series of events, such as for instance an interactive question/answer application.

The basic data structure of the CHM model is a *container*. As will be explained further on, containers are used to structure information in a hypermedia fashion.

To each container is associated a set of zero or more *members*. Each member consists of:

- a *label*: This is a multimedia object. The label defines the content of the member and indicates how the member should be visualized.
- a *container*: This component models the substructure defined by the member, as it consists in turn of a set of members.
- a *ranking*: this is a non-negative integer number, used to determine which members are accessible when a particular member of the container is current.

If a member consists of a label only, then it represents content without further structure: although the multimedia object that acts as a label may possess internal structure, this is not considered in the CHM model, where multimedia objects are 'black boxes'. In this sense, labels are similar to for instance non-HTML documents in the WWW: such documents cannot contain substructure either.

If the member consists of a container only, then its visualisation is not elaborated, but it does define a navigable substructure. This is similar to the status of a menu in Gopher, which can only contain references to submenus or documents and does not contain any multimedia content itself.

If neither the label nor the container are empty, then the label defines the content and the container defines the internal structure of the member.

One of the members is called the *head*. It has a special status as will be further elaborated below.

The function of a member is similar to that of a node in the basic node link paradigm: it acts as the unit of data. The concept of membership replaces that of traditional links, as will be elaborated in the next section.

Membership of containers does not need to be hierarchical: containers can be shared when they belong to members of different containers. Loops, representing recursive relationships, are also allowed.)

2.2 An Example

Consider for instance figure 1: this could be part of a CHM data structure for a Campus-Wide Information System (CWIS). The squares represent containers, the ovals labels. The rankings are indicated between brackets.

In this structure, a professor is modeled by a container with three members:

- two with
  - a label, and
  - a container

  for the courses he teaches and his publications, and
- one with an empty container for his curriculum vitae.

Labels A and B are multimedia objects that visualize the member they belong to in the context of the container $Professor$. Label A could for instance be a video clip of the professor, explaining what his courses are about. Label B can be a list of the more recent of his publications.

Suppose this particular professor teaches two courses, one on databases and another on the Pascal programming language. The container on the database course contains course schedules (label E can for instance be the current schedule, or can indicate where and when the next three lectures take place), reference material for the students, lecture notes, grades (a member probably not freely accessible to everybody, see below) and the practicals submitted by the students (e.g. database schemas they have designed with a CASE tool). The reference material includes a textbook the professor has co-authored with another expert, two standard textbooks and the most influential papers in the field.

One important point is that the container that models the book on databases is *shared* by:

- the container with reference material for the database course, and

– the container with all publications of the professor.

In order to visualize the container *Book on Databases* differently in the two contexts it belongs to, different labels are associated to it in the two containers it belongs to. The label N for instance can adhere to a common format for all publications of this professor, whereas the label J can indicate why this book is relevant as reference material for the course on databases, or for instance which parts of the book are obligatory reading and which parts aren't, etc.

Moreover, there is a *loop* in the data structure: the container on the professor contains a member on his publications, which contains a member on the book, which contains a member on the authors, which contains a member on the professor, etc. (ad infinitum).

Sharing and loops are also illustrated by figure 2, another representation of the CHM data schema of figure 1. Here, a container is linked to his members by directed links from the container to the ranking of the member, from the ranking to its label, and from the label to the container that is part of the member. The loop *Professor – Publications – Book on Databases – Authors – Professor* is evident, as is the fact that the container *Book on Databases* is shared by *Reference Material* and *Publications*. This figure should not give the wrong impression though that the CHM model is equivalent to a hypermedia model based on nodes and labeled links: the arrows in figure 2 do *not* represent links, but rather which members belong to which containers.

3 Navigation

3.1 Introduction

In the CHM model, there is always a *current container* that defines the navigational context. The current container always contains a *current member*. (A container with zero members can never become the current container, see also below.) When a new container becomes current, the head of that container becomes the current member.

3.2 Accessible Members

The CHM model relies on the ranking of members to determine which members are accessible, based on the following rule: *if a member with ranking k is current, then all members with ranking $k-1$, k or $k+1$ are accessible.*

That is why there is a requirement that, for each two rankings k_1 and k_2 with $k_1 \neq k_2$, there must be a member with ranking k for all $k_1 < k < k_2$. This property guarantees that, eventually, all members from a container can be accessed.

As an illustration, consider a container $c = \{(l_1, c_1, k_1), ..., (l_n, c_n, k_n)\}$. In figure 3, different topologies for c are represented, based on different rankings of the members (l_i, c_i, k_i) in c, using the same notation as in figure 1. The arrows point from a member m to all members that are accessible when m is the current member.

In case (a) of figure 3, all members have the same ranking. The result is that, whatever the current member, all members are always accessible. In case (b), the ranking of the members results in a linear sequence: if the current member is not the first (last), then the previous (next) member is accessible. In case (c), all other members are accessible when (l_2, c_2) or (l_3, c_3) is current. When (l_1, c_1) or (l_4, c_4) is current, then only (l_2, c_2) and (l_3, c_3) are accessible. Case (d) can be interpreted along similar lines.

An important consequence of this approach is that *link integrity can be maintained automatically* at all times: in terms of the CHM model, links correspond to membership. Now, if a new member m is added to a container c, then the ranking of m automatically determines when m is accessible. The member m must not be linked manually to the other members of c. Conversely, when m is removed from c, it doesn't need to be explicitly un-linked from the other members of c.

3.3 Access

When `Access` is applied to a member, then that member becomes the current one, and its label (if non-empty) is visualized: this will typically involve the display of multimedia information on the screen, but it might also be a more complex event, e.g. an interactive question and answer application. The function of a label is to carry information about the member, as appropriate within the container that the member belongs to.

Consider e.g. the following container with seven members:

'Multimedia, Hypertext and Hypermedia' =
{ ('A','General Literature on Hypermedia and Multimedia',1),
('B','Hypermedia Data Models',1)
('C','Hypermedia Systems',1)
('D','Hypermedia Concepts',1)
('E','Hypermedia Applications',1)
('F','Multimedia and Hypermedia Standards',1)
('G','Multimedia',1) }

If a user accesses the second member, then multimedia object 'B' is visualized: this could e.g. be an introduction on hypermedia data models and their role in hypertext and hypermedia systems. The label enables a reader to get information about a member, in terms that are relevant within the navigational context of the current container and without leaving that context.

An important point is that the label associated with a destination is *context-dependent.* Consider e.g. the following container:

'Data Modeling' = { ('X','Database Data Models',1),
('Y','Hypermedia Data Models',1)
('Z','Knowledge Representation',1) }

In this container, another label (multimedia document 'Y') is associated to the same container 'Hypermedia Data Models'. This document 'Y' can explain what hypermedia is all about, rather than label 'B' that deals with the relevance of data modeling in the context of hypermedia.

3.4 Zooming In, Out and Up

The navigational context can be changed by applying the Zoom_In operation on the current member. (This is only possible if the current member is non-empty.) The effect of this operation is that the container of the current member becomes the current container. Its head becomes the current member. If the label of that head is non-empty, then it is visualized.

The opposite effect can be achieved by applying the Zoom_Out operation: basically, the situation before the last Zoom_In is reinstated. By zooming in and out, users can focus more or less on the information presented in a particular container.

An important point is that the navigational context of the container is retained when members are accessed. Only when the user zooms in on a member does a new navigational paradigm become active.

Operations Container_Up and Label_Up identify all containers that contain the current container or label in their members. The user can then zoom in on one of those members. This enables a user to explore all contexts a particular container or label participates in.

In the hypermedia data models the authors developed before (the HM model [Andrews, et al. 1995c, Maurer, et al. 1994, Srinivasan, Scherbakov 1995] and the SOS model [Duval, et al. 1995, Duval, Olivié 1995]), similar operations are defined.

3.5 An Example

In order to make the above more concrete, let us consider the following example, an implementation in Home of the two containers mentioned above. If the container Multimedia, Hypertext and Hypermedia is the current container, and (B,Hypermedia Data Models,1) is the current member, then a screen like figure 4 is presented.

The lower part of the screen displays the label B (an introduction on data modeling within a hypermedia context). The middle and upper part display the navigational context: it indicates the current container, the current member (in italics, in the list of accessible members) and the accessible members.

The current member (Hypermedia Data Models) can be zoomed in upon and the accessible members of the current container can be accessed. (The differences between the icons in front of the names are not very clear on figure 4). Suppose for instance the end user zooms in on the current member (Hypermedia Data Models), by clicking on the icon in front of that member in the list of members of the current container. Figure 5 shows the resulting screen.

In figure 5, the container of the current member is Basic Node Link Paradigm, because the author of the container Hypermedia Data Models defined this member as the head. Following the same approach as before, the end user can now zoom in on the current member or access another one.

3.6 Search

As an alternative access paradigm, both labels and containers can be searched for as well, by imposing search constraints on their attributes. Figure 6 e.g. shows the search screen for articles. (These can act as labels.) Search patterns can be defined for any number of attributes and -if the user has the appropriate access rights- the result is a dynamically generated container with a member for each object that satisfies the search criteria. If the object searched for is a multimedia object, then this object will act as a label of a member with an empty container. Otherwise, if containers are searched for, the members of the dynamically generated container will have empty labels.

4 Authoring

4.1 Creation

In this section, we will concentrate on the authoring process in CHM. It is important to note first that content creation, i.e. the design and implementation of multimedia objects, is beyond the scope of this discussion. Multimedia content can be created using a special-purpose authoring system. The result can then act as a label in the CHM model.

The process we discuss here deals with data structuring. This process typically starts with the creation of a container, that will act as a framework for a unit of information. The author then inserts members that include already defined multimedia documents as labels and other containers. In case the latter already exist, they are re-used, possibly in a context different from the one they were originally intended for - as mentioned above, the purpose of the label is exactly to allow integration of existing resources in the context of a new container.

Containers that are part of members can be further elaborated afterwards. In that case, the authoring process proceeds in a top-down way. A bottom-up authoring approach is also possible: in that case, containers are defined and gathered as members of other containers, that can in turn become part of yet other members, etc.

4.2 Deletion

It is important to note that, when a member $M = (A, B, k)$ is removed from a container C, the label A (a multimedia object) and the container B both survive as independent entities. It is only the relationship between A and B, within the context of C, and the relationship between M and the other members of C that are destroyed.

The navigational paradigm of C, as defined by the rankings of its members, is not corrupted when M is removed. This mechanism ensures *link integrity*, as no dangling references can arise. This is very important, especially in a large-scale environment where manual link maintenance is no longer feasible.

When a container C is deleted, all links encapsulated in C cease to exist as well. However, the labels and containers of the members of C survive as independent entities; only their interrelationships in the context of C are destroyed. If C itself belongs to a member $M = (L, C)$ of another container X, then C is replaced in M by a *slot*, i.e. a placeholder for a container (see section 5).

An important point is that *all operations are addressed to one particular container and do not affect other containers.* This leads to self-containment of containers (the more so as navigation is also contained is also encapsulated within a container, as explained in the previous section). Thus, a container is a self-containing module that can be edited and browsed independently of others. This is especially important when a container is re-used as a member of (and therefore in the context of) another container.

4.3 Example

Authoring along the lines discussed above can be carried out with ordinary WWW clients. Note that this is not so when WWW clients access ordinary WWW servers or the Hyper-G gateway [Cailliau 1995]. When the user activates the 'Create/Search' option, then he can define the relevant attribute values (e.g. name, author, etc.), using electronic forms similar to the one of figure 6.

Activation of the 'Add Member' option leads to a number of screens, in order to select the label (out of all available multimedia objects) the container to associate with the label, the container that the member will belong to and the ranking of the member.

5 Slots

The final fundamental notion of the CHM model is a *slot*, which is a placeholder for a container in a member. Slots are useful when an author wants to refer to a container that doesn't exist yet or anymore.

As an example of the first case, consider e.g. an author who wants to create information about database management systems. He can create different containers for such systems based on the hierarchical, network, relational and object-oriented data models. Following a top-down approach, he can first create the general container DBMS, with four members that have slots where the yet-to-be-authored containers for the four different models will fit:

```
DBMS = {(H,slot,1),(N,slot,1),(R,slot,1),(O,slot,1)}
```

where H, N, R and O represent labels that carry information about the hierarchical, network, relational and object-oriented models respectively.

As an example of the case where a slot is a placeholder for a container that doesn't exist any more, consider the case where the author re-uses a container on relational databases that has been elaborated by another author. Suppose that, for whatever reason, the latter author destroys his container. In that case, a slot will remain in the member that the now destroyed container belonged to. The author of the DBMS container can fit another container on relational databases in the slot, if and when such a container becomes available.

It is important to understand how slots and labels interrelate: when a member contains a slot, the label carries information about the sort of information that will be presented in the container that fits in the slot, and about the reason why this container can be a member of the encompassing container.

Another important point is the difference between a member

- with a label and no container, used to include multimedia information only (the label), and no further hypermedia structure;
- with a label and a slot, for the case where the container that will fit in the slot doesn't exist yet or any more;

- with a label and an empty container, in which case the container just happens to be empty (but members can be inserted into it at any time).

6 Some implementation details

As mentioned above, we have implemented the CHM model in the Home environment for management of distributed hypermedia objects [Duval, et al. 1995, Duval, Olivié 1995], although, currently, slots are not fully implemented.

Home includes a layer for multimedia data management. The content of multimedia objects is referred to by Universal Resource Locators, as defined in the WWW [Berners-Lee, et al. 1994, Cailliau 1995]. These multimedia objects can act as labels in the CHM implementation of Home. So, any document accessible over the WWW can act as a label in Home. It doesn't need to be stored in the Home server.

On the lowest relevant layer of Home, data are managed using a commercially available Relational DataBase Mangement System (RDBMS), in casu Oracle. As explained in [Duval, Olivié 1995, Duval, et al. 1995], using a DBMS for data management, rather than directly relying on the file system (as in regular WWW servers), results in extra functionality (ACID properties of transactions, access rights and user groups, query engine, etc.).

The software that implements functionality on top of the Oracle RDBMS has been developed in Tcl [Ousterhout 1994], using the OraTcl extension for interaction with the Oracle database.

Home is accessible to WWW clients through our WWW server, using the Common Gateway Interface (CGI) mechanism of the latter server to gateway requests from the WWW client to the Home server. The same mechanism is used to deliver the results, packaged in the form of dynamically generated HyperText Mark-up Language (HTML) documents.

The Home server can also be accessed in a more direct way, without passing through the WWW gateway. As an example, we have developed an authorinhg client in Tk [Ousterhout 1994].

As a partial summary of the previous section, three key points indicate the main added value of our server when compared to a regular WWW server:

- *link maintenance* is an automatic process, as a member will no longer be accessible when it is removed from a container; compare this situation with the frequent phenomenon of dangling references in the WWW;
- *search facilities* are fully integrated and offer a complementary access paradigm to navigation over hyperlinks;
- *update facilities* are available through any WWW client that supports electronic forms, whereas the standard WWW mode is read-only [Cailliau 1995].

For the above reasons, Home can be considered as a second-generation WWW server [Andrews, et al. 1995b].

7 Previous and Related Work

7.1 Previous Work

As explained in [Duval, et al. 1995], hypermedia facilities in Home used to be based on a very simple data model, called SOS, for "Sets Of Sets". This model is similar to the data model supported by the Gopher system [Obraczka, et al. 1993]. In fact, the SOS model corresponds to a CHM model without labels or slots. The result is a hierarchy of containers (possibly with sharing), which is probably quite appropriate in many database-like applications, but not when more advanced facilities are required (like e.g. in Computer-Aided Learning applications [Andrews, et al. 1995c, Hendrikx, et al. 1995, Srinivasan, Scherbakov 1995]).

The main difference with the HM model [Srinivasan, Scherbakov 1995] is the introduction in CHM of labels that describe the member they belong to in terms appropriate for the context of the current container. In the HM model, when a member is accessed, its head [Srinivasan, Scherbakov 1995] or label [Andrews, et al. 1995c] (a multimedia object associated to the member, whatever container it belongs to) is visualized. This means that the visualization of a member is fixed and cannot be adapted to the contexts of the different containers it belongs to. As mentioned before, labels should facilitate re-use of existing resources in new contexts.

7.2 Related work

A number of other data models are similar to ours, particularly because they are also set based. These models only define data structuring facilities though, without the behavioural component that must be addressed in an object oriented approach.

- In *Hyper-G* [Andrews, et al. 1995a, Kappe, et al. 1993], documents can be grouped into collections. The latter can in turn belong to other collections. One of the members of a collection can be designed as head and will then be displayed automatically when the collection is accessed or zoomed in upon. A Hyper-G collection is somewhat similar to our container concept, but there is no construct in Hyper-G that corresponds to our notion of a context-dependent label. Moreover, hyperlinks can refer to other documents across collection boundaries, so that there is no self-containment of collections. This makes re-use of resources more difficult.
- In [Garzotto, et al. 1994], 'collections' (sets with an inner navigable structure) are added as a fundamental structuring unit to the Dexter model (see also below). Atomic values, called 'slots', are grouped in 'nodes'. The latter act as units of navigation, and can in turn be grouped into a collection. Roughly speaking, a slot in this model corresponds with a multimedia object in CHM. Nodes and collections both correspond with our notion of containers. In fact, we believe the distinction between slot (or rather, a node with one slot automatically created for each atomic value) and node is unnecessary and results in conceptual complications.
  The node associated to a collection in [Garzotto, et al. 1994] is similar to our notion of label, but the latter is context dependent, whereas the former is fixed for a collection. Also, [Garzotto, et al. 1994] defines no `Access`, but only a `Zoom_In` operation. Therefore, when a new collection becomes current, the existing navigational context is lost.

Other, non set based, approaches for advanced hypermedia modeling include e.g.:

- The *Dexter* model [Halasz, Schwartz 1994] is a formalized representant of the basic node link paradigm. The *storage layer* manages persistent components, that include a content, a set of attributes, a presentation specification an a set of anchors. Composite components reflect 'is part of' relationships. In the Dexter model, composites can only contain base components, which correspond to data objects, and cannot contain other composites.
- [Gronbaek 1994] generalizes the Dexter composite mechanism, in the context of the Dexter based *DEVISE Hypermedia* framework (DHM). An important difference with our notion of containers is that components 'do not know about which and how many composites they are members of'; hence, no equivalent to the `Zoom_out` operation in DHM. Moreover, the idea of local containment, both with regard to navigation and authoring, so important for re-use, is not addressed in DHM.
- *Microcosm* is based on an approach radically different from ours: navigation is controlled by a filtering mechanism [Hill, Hall 1994]. User made selections of documents (comparable to the traditional notion of anchors) can be sent to linkbases that deliver the corresponding destination documents.

8 Future

We have a number of ideas for further developments and applications of the Home system. Some of our more concrete plans for the future include:

- As explained above, membership of containers is extensionally defined, by manually inserting members, or by filling in slots. An alternative mechanism can rely on *intensional* membership, when a set of search criteria defines the members of a container. This mechanism is currently supported, but definition of intensional memberships cannot be achieved through ordinary WWW clients yet. Instead, the search criteria must be defined through a Tcl script that accesses the Oracle DBMS through an embedded SQL query.
- The data structuring facilities of our model can be used to define *data schemas*, that are an instantiation of the CHM model for a particular application domain, as is customary in the field of database design. On the one hand, a data schema based approach can be contradictory to the often much appreciated flexibility of the hypermedia approach; on the other hand, it can help to guarantee consistency of data structuring in large scale environments, such as e.g. a Campus-Wide Information System (CWIS). Although some research into this area has been carried out, we believe that this issue certainly deserves further investigation.
- We are currently developing some applications on top of Home. These include a self study course on basic computer science [Hendrikx, et al. 1995] and a project for European collaboration on courseware re-use. We are also investigating the possible use of Home in an information system on architecture and a Campus-Wide Information System.

9 Conclusion

Above, we have presented a new data model for hypermedia. Our model, called CHM, for Contained HyperMedia, follows a set based approach and supports self-containment of data units, both with regards to navigation and authoring. We believe our approach lends itself well for re-use of existing resources in a new context. The discussion of our model not only dealt with data structures, but also presented in some detail the operations for authoring and navigation. We have designed and implemented this model in the context of the Home system for distributed hypermedia management. Because the latter system is accessible to WWW clients, it can be considered a WWW server of the next generation, offering improved support for authoring and automatic link maintenance.

Acknowledgements

We wish to name explicitly Koen Hendrikx and Rudi Maelbrancke, both colleagues working at the computer science department of the K.U.Leuven, for the numerous discussions and their continued support. We also gratefully acknowledge the partial financial support provided by the Belgian National Fund for Scientific Research.

References

[Andrews, et al. 1995a] K. Andrews, F. Kappe, and H. Maurer. Serving information to the web with hyper-g. In *WWW95: Third International World-Wide Web Conference*, April 1995. Available from http://www.igd.fhg.de/www/www95/papers/105/hgw3.html.

[Andrews, et al. 1995b] K. Andrews, F. Kappe, H. Maurer, and K. Schmaranz. On second generation network hypermedia systems. In H. Maurer, editor, *Educational Multimedia and Hypermedia, 1995. Proceedings of ED-Media 95, World Conference on Educational Multimedia and Hypermedia*, pages 75–80, June 1995. Available from http://www.iicm.tu-graz.ac.at/Cedmedia.

[Andrews, et al. 1995c] K. Andrews, A. Nedoumov, and N. Scherbakov. Embedding courseware into the internet: Problems and solutions. In H. Maurer, editor, *Educational Multimedia and Hypermedia, 1995. Proceedings of ED-Media 95, World Conference on Educational Multimedia and Hypermedia*, pages 69–74, June 1995. Available from http://www.iicm.tu-graz.ac.at/Cedmedia.

[Berners-Lee, et al. 1994] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The world-wide web. *Communications of the ACM*, 37:76–82, Aug 1994.

[Cailliau 1995] R. Cailliau. About www. *Journal of Universal Computer Science*, 1:221–230, Apr 1995. Available from http://www.iicm.tu-graz.ac.at/Cjucs_root.

[Conklin 1987] J. Conklin. Hypertext: An introduction and survey. *IEEE Computer*, 2(9):17–41, September 1987.

[Duval, et al. 1995] E. Duval, H. Olivié, P. O'Hanlon, and D. G. Jameson. Home: an environment for hypermedia objects. *Journal of Universal Computer Science*, 1:265–287, May 1995. Available from http://www.iicm.tu-graz.ac.at/Cjucs_root.

[Duval, Olivié 1995] E. Duval and H. Olivié. A home for networked hypermedia. In H. Maurer, editor, *Educational Multimedia and Hypermedia, 1995. Proceedings of ED-Media 95, World Conference on Educational Multimedia and Hypermedia*, pages 193–198, June 1995. Available from http://www.iicm.tu-graz.ac.at/Cedmedia.

[Garzotto, et al. 1994] F. Garzotto, L. Mainetti, and P. Paolini. Adding multimedia collections to the dexter model. In *Proceedings of ECHT 94: European Conference on Hypermedia Technology*, pages 70–80, September 1994.

[Gronbaek 1994] K. Gronbaek. Composites in a dexter-based hypermedia framework. In *Proceedings of ECHT 94: European Conference on Hypermedia Technology*, pages 59–69, September 1994.

[Halasz, Schwartz 1994] F. Halasz and M. Schwartz. The dexter hypertext reference model. *Communications of the ACM*, 37:30–39, Feb 1994.

[Hendrikx, et al. 1995] K. Hendrikx, E. Duval, and H. Olivié. Hypermedia for open and flexible learning. In *WCCE95: Sixth IFIP World Conference on Computers in Education*, July 1995.

[Hill, Hall 1994] G. Hill and W. Hall. Extending the microcosm model to a distributed environment. In *ECHT94: ACM European Conference on Hypermedia Technology*, pages 32–40, September 1994.

[Kappe, et al. 1993] F. Kappe, H. Maurer, and N. Scherbakov. Hyper-g. a universal hypermedia system. *Journal of Educational Multimedia and Hypermedia*, 2:39–66, 1993.

[Maurer, et al. 1994] H. Maurer, N. Scherbakov, K. Andrews, and P. Srinivasan. Object-oriented modelling of hyperstructure: overcoming the static link deficiency. *Information and Software Technology*, 36:315–322, 1994.

[Nielsen 1990] J. Nielsen. *Hypertext and Hypermedia.* Academic Press, 1990.

[Obraczka, et al. 1993] K. Obraczka, P. B. Danzig, and S. Li. Internet resource discovery services. *IEEE Computer*, 26(9):8–22, September 1993.

[Ousterhout 1994] J. K. Ousterhout. *Tcl and the Tk toolkit.* Addison-Wesley, 1994.

[Parunak 1991] H. Van Dyke Parunak. Don't link me in: Set based hypermedia for taxonomic reasoning. In *Hypertext'91 Proceedings*, pages 233–242. ACM, December 1991.

[Srinivasan, Scherbakov 1995] P. Srinivasan and N. Scherbakov. Embedding infer engines into educational hypermedia. In H. Maurer, editor, *Educational Multimedia and Hypermedia, 1995. Proceedings of ED-Media 95, World Conference on Educational Multimedia and Hypermedia*, pages 603–608, June 1995. Available from http://www.iicm.tu-graz.ac.at/Cedmedia.

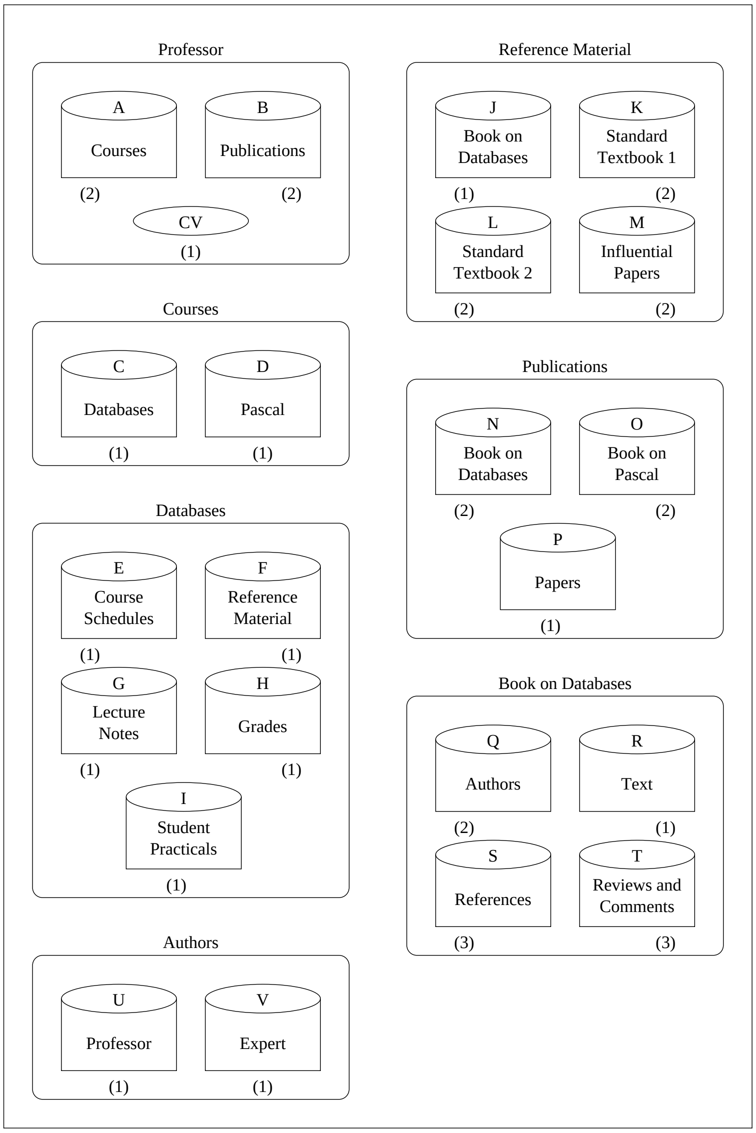

Figure 1: A CHM hypermedia corpus for a CWIS

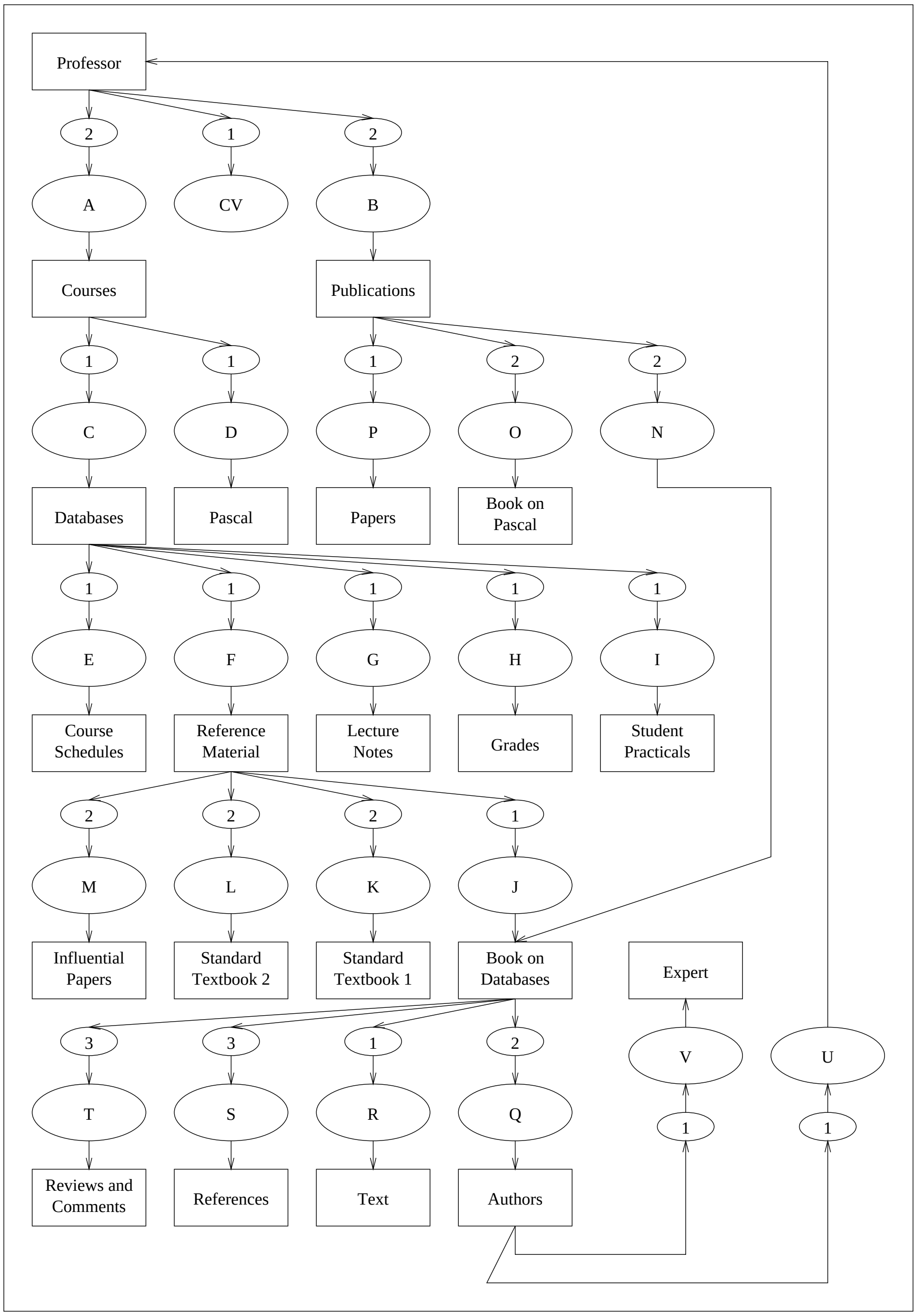

Figure 2: An alternative representation of the data schema

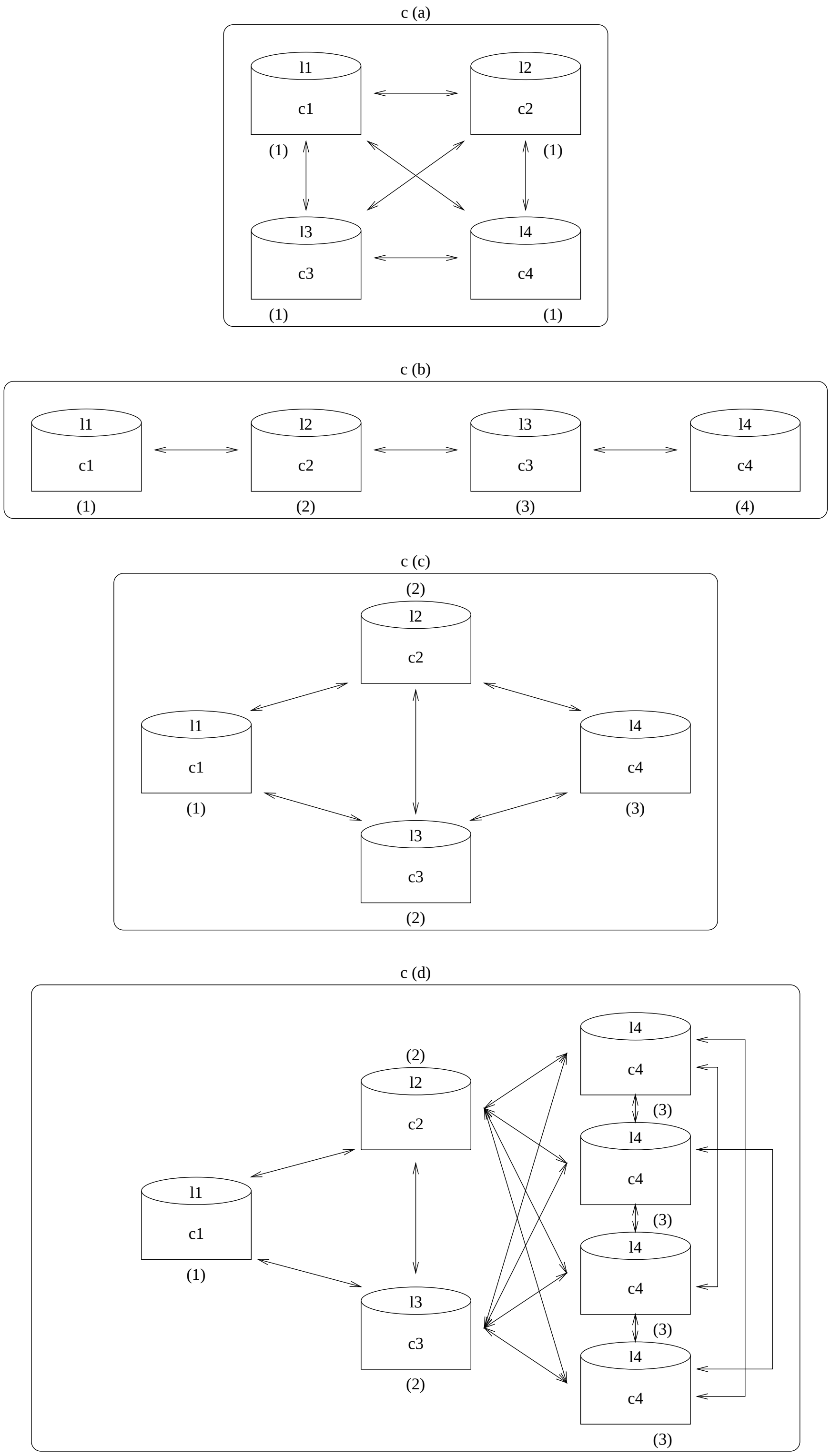

Figure 3: Accessible Members

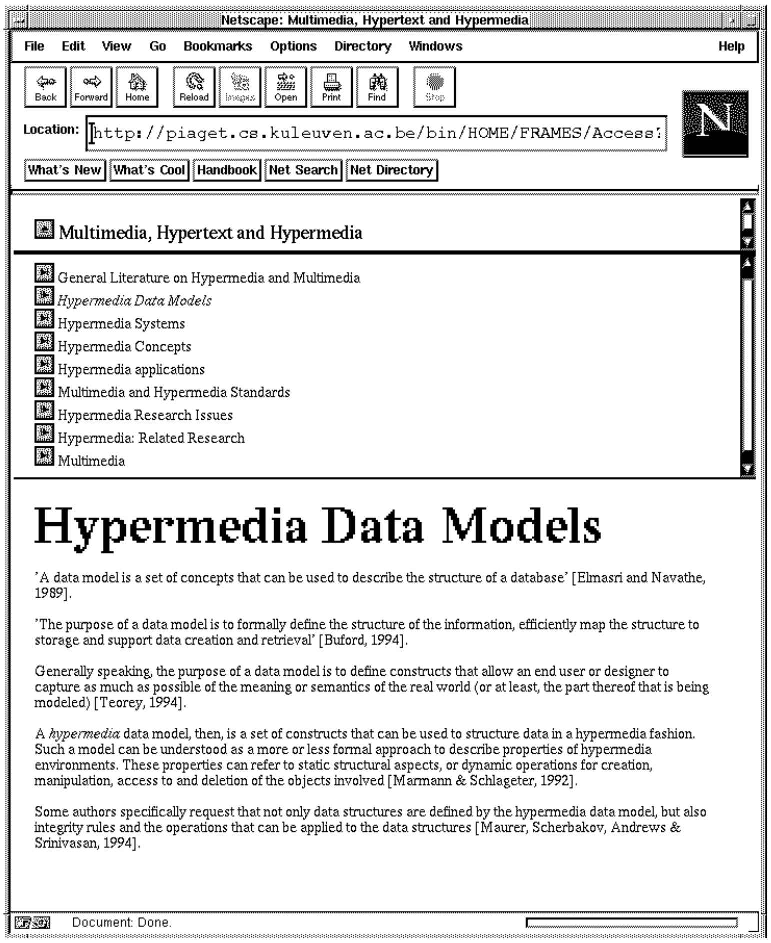

Figure 4: Current container is Multimedia, Hypertext and Hypermedia

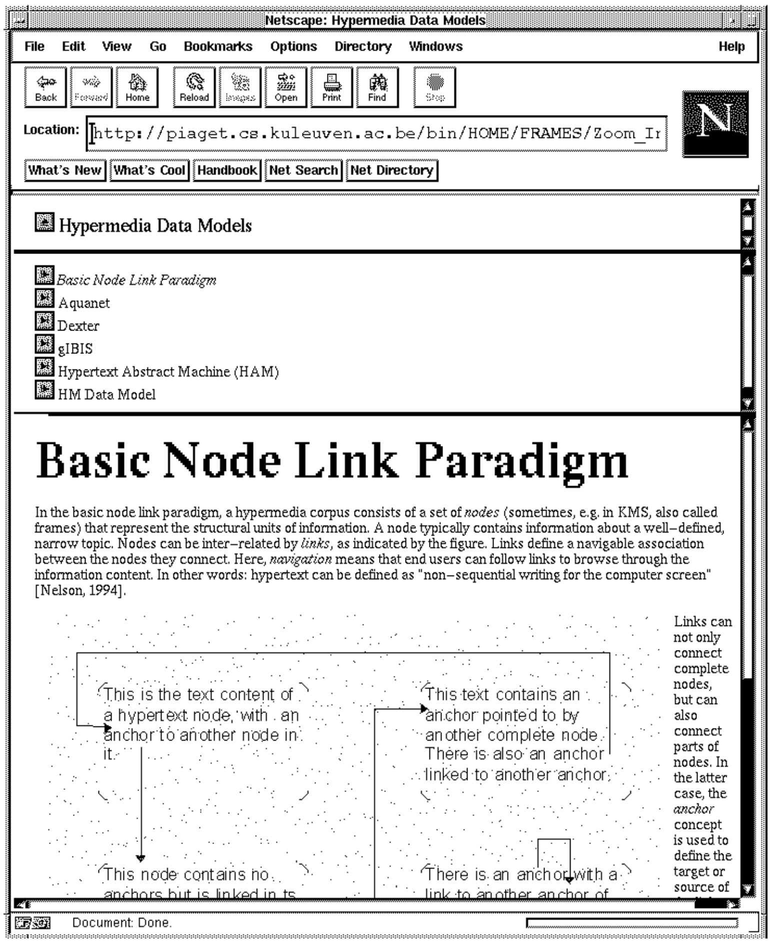

Figure 5: Current container is Hypermedia Data Models

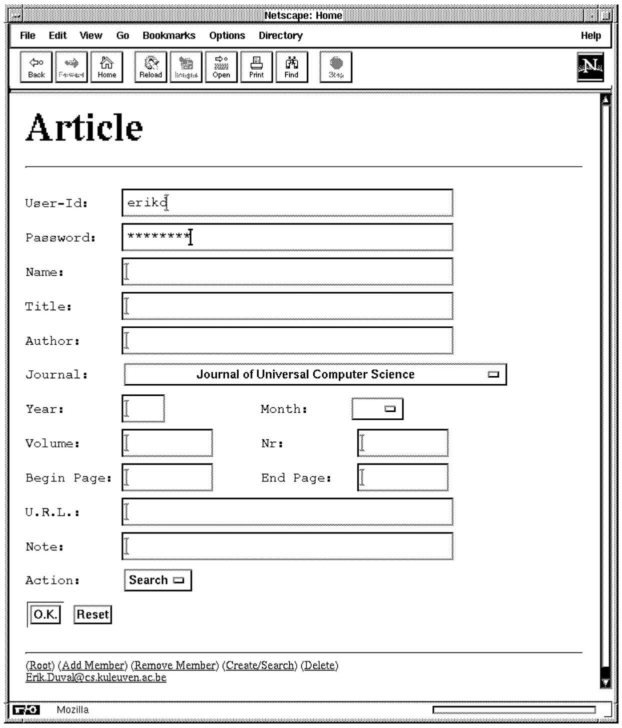

Figure 6: Client screen for searching