# Design and Maintenance of Web-Based Information Systems

Paolo Merialdo

March 9, 1998

# Contents

# List of Figures

# Chapter 1

# The ARANEUS Methodology: Overview

The ARANEUS Web design methodology is a thorough and systematic design process to design the organization of large amounts of data on a Web hypertext. In this introductory chapter, we discuss the major features of our approach. First, in Section 1.1 we discuss the importance of introducing several levels in the description of hypertexts; thus, in Section 1.2 we address general issues in generating a Web hypertext; therefore, in Section 1.3 we illustrate how the hypertext design consists of several phases; finally, in Section 1.4 we conclude with a discussion on the state of arts.

## 1.1 Hypertext Description Levels

It is now widely accepted that essentially every application needs a precise and implementation independent description of the data of interest, and that this description can be effectively obtained by using a database conceptual model, usually a version of the Entity-Relationship (ER) model [14]. Since most hypertexts offered on the Web, and especially those we are mainly interested in, contain information that is essentially represented (and stored) as data, our methodology starts with conceptual database design and uses the conceptual scheme also as the basis for hypertext design (following previous proposals in this respect, in particular RMM [36]). At the same time, departing in this from existing approaches, we believe that the distance between the ER model (or any other conceptual data model), which is a tool for the representation of the essential properties of data in an abstract way, and HTML (or any other language/model for hypertext implementation) is indeed great. There are in fact many types of differences.

A first important point is that a conceptual representation of data always separates the various concepts in a scheme (this is the conceptual counterpart to database normalization), whereas in hypertexts it is reasonable to show distinct concepts together (denormalizing or adding nested structures, in database terms), for the sake of clarity and effectiveness in the presentation.

Moreover, a conceptual model has links between entities only when there are semantically relevant (and usually non-redundant) relationships, whereas hypertexts usually have additional links (and nodes) that serve as access paths. Specifically, each hypertext has an entry point (the home-page, in Web terminology), and other additional pages that are essential for the navigation.

Also, relationships in the ER model are undirected, whereas hypertextual navigation is conceptually directed (often, but not always, bidirectional). Additional issues follow from the way collections of homogeneous entities are actually represented in hypertexts: by means of sets of similar pages or by means of one page with a list of homogeneous elements.

Then, there are specific issues related to features of the hypertext language (HTML, in our case) that could be useful to represent at a level that is more abstract than the language itself but too detailed for being relevant together with the initial descriptions of links. For example, a list could suffice to access objects of a certain class, if there are few instances, whereas for a class having several elements a direct access should be more appropriate: HTML has the form construct that could be useful in this respect. This distinction is clearly not relevant at the conceptual level, but it is certainly important to specify it well before going down to HTML code.

Finally, there are features that are related only to the presentation of data, and not to their organization: the actual layout of an HTML page (or a homogeneous set thereof) corresponds to one of the possible "implementations" of the logical structure of the involved data.

Our methodology takes these issues into account by offering three different levels (and models) for the definition of hypertexts, by separating the various features, from the most abstract to the concrete ones. At the highest level of abstraction we have the *hypertext conceptual* level, rather close to the database conceptual level: hypertexts are defined by means of the *Navigation Conceptual Model* (NCM), a variant of the ER model inspired by RMM [36]; beside ER concepts, it allows the specification of access paths (possibly with additional nodes) and directions of relationships as well as nested reorganizations of data. Then, we have the *hypertext logical* level, where the data contained in the hypertext is described in terms of pages and their types (*page schemes*); here we use the ARANEUS *Data Model* (ADM) we have recently proposed [11]. Finally, the organization of data in pages, the layout, and the final appearance are issues that do not influence data except in the presentation. Therefore we propose that there is a *presentation level* concerned with HTML templates (prototypical pages) associated with page schemes.

## 1.2 Generation of Web sites

Since pages are organized according to the ADM scheme, if data are stored in a database, as we believe it should often be (and in fact it is) the case, the construction of the Web site is almost completely determined by the ADM scheme itself and the page templates that form the presentation level description of the hypertext. Indeed, a mapping between data in hypertext pages and values in the database can be established, in order to automatically generate actual HTML pages. This can be attained in a number of ways. At least two different choices are possible here [49]: each page can either *(i)* be generated off-line starting from the database, materialized and stored in an HTML file (the *push* approach), or *(ii)* be dynamically generated upon request (the *pull* approach). The push alternative has clear advantages in terms of performance, since it cuts database access costs, but requires to update the hypertext periodically to reflect database changes.

Several commercial database systems [2, 4] now provide functionalities for the automatic generation of pages. However, they mainly allow for dynamically generating a single page at a time, containing a set of database tuples (the pull approach).

In our proposal, the mapping from the hypertext to the database is based on the hypertext and database logical schemes. A specific programming language, called PENELOPE [11] can be used to this end. PENELOPE programs automatically generate HTML pages based on the ADM scheme and on the associated templates. PENELOPE supports both push and pull solutions: it can generate and materialize hypertext pages starting from the database content, or be used to dynamically generate pages using CGI-scripts. Moreover, it also allows for intermediate solutions, in which some of the pages are materialized, and some others are dynamically generated upon request. PENELOPE also simplifies site maintenance: updates to the underlying database are directly reflected on the site and URLs are kept consistent also in the presence of page insertion or deletion (this is based on the specific URL invention mechanism used by PENELOPE, borrowed from object-oriented databases [33]).

## 1.3 The phases of the ARANEUS Design Methodology

The issues discussed above motivate the organization of the ARANEUS Design Methodology. Figure 1.1 show the phases, the precedences among them, and their major products. Let us briefly comment on each of them.

Given the central role data have in the Web sites we consider and the maturity of database methodologies [14], our Phases 1 and 2 involve the standard

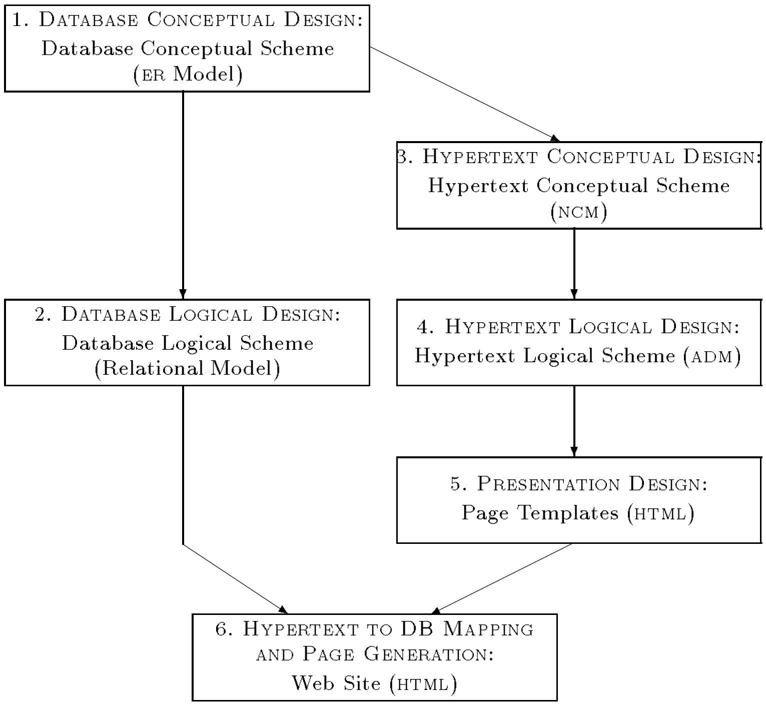



Figure 1.1: The ARANEUS Design Methodology

conceptual and logical design activities for databases. Conceptual and logical database schemes, beside being used to implement the database (whose physical design activity is needed but omitted from the figure since it is not relevant here), are also used as input for hypertext design and implementation. More precisely, the database conceptual scheme (according to a version of the ER model) is the major input to Phase 3 (*hypertext conceptual design*), where it is "transformed" into a hypertext conceptual scheme, in our *Navigation Conceptual Model* (NCM). Then, Phase 4 (*hypertext logical design*) receives an NCM scheme as input and produces an ADM (logical) scheme. Phase 5 (*presentation design*), given the ADM scheme, associates an HTML template with each of its page schemes. Finally, Phase 6 (*hypertext to database mapping and page generation*) makes use of the logical database scheme (produced in Phase 2) and of the hypertext logical scheme and the associated templates, in order to generate actual HTML pages.

The organization of the methodology is modular, since the phases interact only via the respective products. This means that it is possible to adapt the methodology to specific contexts: for example, although we proceed as if the database and the hypertext are designed in parallel, it may be the case that the database already exists, and so Phases 1 and 2 are not needed (assuming that the conceptual scheme exists, otherwise a reverse engineering activity could be needed). Also, the methodology can be profitably adapted to support maintenance activities, especially if the modifications concern only the hypertext: the conceptual and logical description of the site represent an essential documentation, based on which the overall quality of the chosen structure can be evaluated, both in terms of effectiveness and performance, possibly allowing for re-organizations.

## 1.4 Related work

Various methodologies have been presented in the context of hypermedia design, including HDM [30, 29], RMM [36] and OOHDM [42]. All of them introduce models for the description of hypermedia applications, the essential constructs being the ones of *entity*, *link* and *menu*, the latter used to represent access structures. Also, we observe that RMM is used as a design methodology also by other works which are concerned with specific aspects of the design process, such as site deployment over the network [41] or the interaction with the site [44].

Both RMM and OOHDM organize the design process in specific phases: *(i) conceptual data design*, i.e., a conceptual description of the domain of interest, based on ER or on object-oriented data models; *(ii) navigation design*, based on specific data models; *(iii) interface design*; *(iv) implementation.* ARANEUS builds on these proposals, with several differences. First, we see the design process as the result of a strict interconnection between two separate activities, database design and hypertext design; moreover, as discussed above, we clearly distinguish the conceptual aspects of the hypertext from logical aspects, whereas the data models used in [36, 42] mix together conceptual constructs, such as *entity* or *directed relationship*, with logical (or even "physical") aspects, such as *guided tours* or *indexed tours*; finally, we specifically focus on *maintenance* aspects and try to provide tools supporting site reorganizations.

The distinction between conceptual and logical aspects is essential in our approach. In fact, besides our Navigation Conceptual Model (NCM), which aims at describing conceptual properties of the hypertext, we use a specific page-oriented data model for the description of the site even at a *logical* level, which is absent in other proposals. The basis of our logical data model–the ARANEUS data model–is the notion of page-scheme, which allows for the description of pages with a uniform structure. This approach has several advantages, since allows designers to specify the organization of data in Web pages, with a clear separation between this issue and the graphical presentation of data. For example, in RMM it is not possible to specify whether each instance of an entity should correspond to a single page or all instances should be in a single page. Also, it is difficult to reason about performance issues, since, for example, there is no notion of *form*, a construct very common in Web sites. Moreover, the absence of a concise description of the page structure makes re-structuring initiatives more difficult.

The ARANEUS data model can be considered as a subset of ODMG [19], in the sense that the notion of page-scheme can be assimilated to the notion of class. However, there are some important differences, motivated by the nature of HTML documents: first, there is only one collection type in ADM, namely, lists; moreover, inheritance is not present in ADM, whereas heterogeneous union is supported; also, identifiers in ADM–that is URLs–are visible to the user and can be treated like any other value; finally, ADM provides a *form* construct, which is specific of the Web framework.

A notion of scheme similar to the one introduced in ARANEUS has been recently used in WGLog [23], whose aim is at studying graph-based query languages for the Web, and in WAG [18], which also studies mining and integration problems in the Web framework.

In HDM [30, 29], two different activities are considered: *authoring in the large* and *authoring in the small*. Authoring in the large aims at describing the overall organization and behavior of a hypermedia application, whereas authoring in the small deals with specific details in the application. HDM mainly focuses on authoring in the large, and only some constructs (*node type, frame, slot*) are given for authoring in the small; however, this is not sufficient for a complete description of a page structure, since, for example, there is no complex data type to model lists or nested lists inside pages.

Fraternali and Paolini have recently developed AUTOWEB [28], a system and a methodology to implement Web sites. AUTOWEB uses a "light" HDM data model to specify a conceptual scheme of navigation, which is the input to automatically produce a database scheme. Deciding the organization of data in the site is an activity supported by a specific design methodology; based on this design phase, data stored in a relational database is translated into HTML pages.

A recent proposal for the management of Web sites comes from the STRUDEL system [25, 27, 26], which aims at applying concepts from database management systems to the process of building Web sites. STRUDEL shares with us the key idea of separating the management of the site's data, the creation and management of the site's structure, and the graphical presentation of the site's pages. The data model underlying STRUDEL is a semi-structured model [8, 17, 16, 20, 9], based on labelled directed graphs. This model is used to declaratively define the Web site's structure by means of STRUQL, a query and transformation language. The result of evaluating a STRUQL query is a site graph which, due to its semi-structured data model, represents both the site content and structure.

The area of languages for Web-site generation is very fertile. In the framework of the RIO project [47], Paradis et al. present a *Prescription Language* [39, 48] for writing documents by restructuring information from various data sources. Also these proposals mainly adopt a graph-based model, in the spirit of OEM [20, 9], and have no notion of schema of a site. A similar approach is that of the YAT system [22, 43], which deals with the problem of implementing a Web site as a view over a set of data sources.

The main differences of these proposals with ARANEUS deals with the the choice of the semi-structured data model as the basis for a Web repository. Moreover, the above systems do not allow a dynamic generation of Web pages, supporting only the push approach.

The motivations in favour of a semi-structured data model are discussed by Fernandez et al. in [26], where they focus on two major points. First, they argument that the labeled direct graph data model is appealing for the Web, viewing each Web site as a graph of pages. The second reason they discuss deals with the advantages that arise from a semi-structured data model in facilitating the integration of data from multiple, non-traditional sources.

However, in our opinion, the Web-as-a-graph approach, may be effective to provide a model for the Web as a whole–for querying purposes. On the contrary, in the management of large data intensive Web sites, in order to assist designers and site administrators in their activities, we argue that a data model should be able to catch regularities, as well. We have experienced that ADM is enough flexible to model the exceptions that may occur in the management of large Web hypertexts, and profitably takes into account the logical organization of data in uniform pages, at the same time. Moreover, since our aim is to design Web site as large enterprise information systems (as HCISs are), we need to leverage on reliable and effective technology for data management: this is not the case for semi-structured repositories, which have enormous lacks in performances.

Several commercial database systems (see, for example, [4, 2, 1]) now provide functionalities for the automatic generation of pages. However, also in that case, no data model is used to describe pages and hypertexts. Moreover, these proposals tend to adopt a pull approach to Web publishing, whereas we also support materialized approaches.

# Part I

# Data Models

This part deals with the formalisms we adopt to describe hypertexts: as we argued in the previous chapter, we use the *Navigation Data Model* (NCM) and the *Araneus Data Model* (ADM) in order to describe Web hypertexts at different levels of abstraction.

A high-level representation of a hypertext concerns the entities of interest (real-world objects to be represented in the hypertext), the relevant paths among them, and the additional access structure. These issues are at the basis of NCM, our data model for the *conceptual* description of hypertexts. ADM acts at a logical level: its constructs give support for describing the logical organization of data in pages, which represent the main means to arrange information in a Web hypertext.

Several examples are used thorough the discussion in order to better explain the presented concepts: in Chapter 2 we discuss NCM by means of examples concerning to a University Clinic; in Chapter 3, we illustrate ADM using examples that refer to the Web interface of the MrBrAQue system, and to OncoLink [3], a bibliographic service provided by a Web site at the University of Pennsylvania.

# Chapter 2

# The Navigation Conceptual Model

We consider a hypertext as a vehicle to present the data relevant to a given universe of discourse: the main classes of objects are organized in nodes, and links provide navigation paths to browse information. In this perspective, a *conceptual data model of navigation* aims at giving basic constructs to describe how concepts from the application domain fit with the organization of information in a hypertext. By means of a conceptual data model of navigation, hypertext designers describe both the main classes of objects of the application domain and the relevant navigation paths between them, at a high level of abstraction.

In hypertexts there exist two kinds of navigation path. On the one hand, we have paths that come from conceptual associations between different classes of object; for example, consider a link that connects pieces of information concerning a given physician to personal data of a patient such a physician is caring: this connection derives from the conceptual relationship between physicians and their patients. On the other hand, a different kind of paths come from the access structure of hypertexts, which is usually based on a hierarchical aggregation of classes of objects and has the function of providing paths to access information. In the Web framework, a typical example is the home page: it aggregates links to access the content of the site.

In this chapter, we present NCM, our data model for the conceptual description of hypertexts. NCM is inspired from the ER data model: it tailors ER constructs such as entities and relationships in the hypertext framework, and introduces new tools to describe the access structure of the hypertext. The chapter proceeds as follows: first, we present macroentities and directed relationships, which are used to give a hypertextual view of the application domain, and could be considered as extensions of ER entities and relationships in the hypertext framework; thus, we introduce aggregations, which allow for a conceptual description of the hypertext access structure; finally, we discuss union nodes and roles: they are NCM constructs that catch particular aspects of hypertexts, dealing with the organization of information and with the navigation paths.

## 2.1 Macroentities

Macroentities are intensional descriptions of classes of real world objects to be presented in the hypertext. They indicate the smallest "autonomous" pieces of information that have an independent existence in the hypertext. Macroentities are the NCM counterpart to ER entities, because of the common correspondence to real-world objects, nevertheless some important differences with respect to ER entities arise. In fact, macroentities have to be relevant from the hypertextual point of view, in the sense that they are used in order to describe hypertext elements: each element should provide pieces of information that suffice for a complete description of the element itself. This leads for example to introduce redundancies – the same piece of information may occur in several macroentities – and violations of the "normal-form", which should not be the case for ER entities.

For a Web hypertext dealing with pieces of information about research in a University Clinic, example of macroentities could be `PHYSICIAN`, and `RESEARCH-GROUP`.

Graphically we represent macroentities by means of rectangles, as shown in Figure 2.1 for `PHYSICIAN` and `RESEARCH-GROUP`.

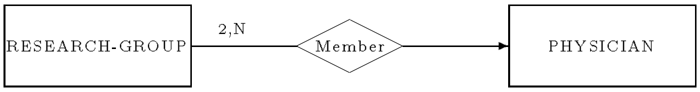



Figure 2.1: Macroentities `PHYSICIAN` and `RESEARCH-GROUP`, and the directed relationship `Member`

## 2.2 Directed Relationships

A directed relationship describes how it is possible to navigate to a destination macroentity from one or more source macroentities, on the basis of a conceptual association.

### 2.2.1 Binary Directed Relationships

Binary directed relationships have a source node and a destination node; *cardinality constraints* specify the minimum and maximum number of instances of the destination node associated with one instance of the source node. We also have *symmetric* directed relationships, that can be seen as composed by two asymmetric directed relationships, being one the inverse of the other; they are used to indicate that navigation between the two nodes can proceed in both ways.

In the graphical representation, diamonds symbolize directed relationships, and an arrow entering the destination node describes the direction of traversal; a labelled edge connects the source macroentity with the diamond: the label is a pair of values specifying minimum and maximum cardinalities. In Figure 2.1 the directed relationship `Member` connects `RESEARCH-GROUP` to `PHYSICIAN`: the arrow imposes that navigation is allowed only from the former to the latter macroentity, and the cardinalities specify that from each research group it is possible to reach at least two physicians. Figure 2.2 shows the symmetric directed relationship `Responsible`: one can navigate both from a given physician to his/her patients, and from a given patient to his/her responsible physician.

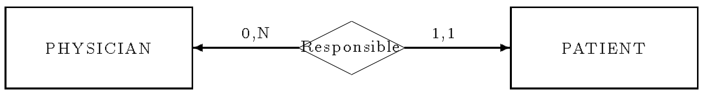



Figure 2.2: A symmetric directed relationship

#### Recursive Directed Relationships

Recursive directed relationships connect a macroentity to itself. For example, the directed relationship `Supervisor` in Figure 2.3 connects a physician to his/her supervisor, both represented by the macroentity `PHYSICIAN`.

### 2.2.2 N-ary Directed Relationships

N-ary directed relationships are associations that involve N macroentities among which at least one plays the role of destination node. Each destination node is associated with N-1 binary navigation paths, which, at the instance level are links connecting instances of the each source node to instances of the destination node.

For each source macroentity cardinalities have to be specified: given a destination node, among the macroentities participating the association, cardinality constraints express the minimum and maximum number of instances

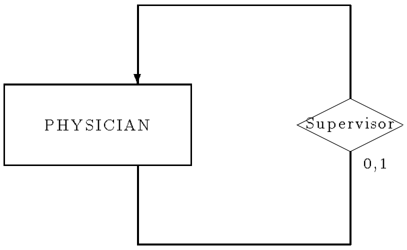



Figure 2.3: A recursive directed relationship

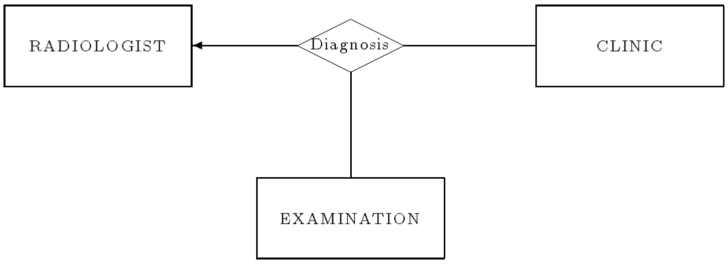



Figure 2.4: Macroentities `RADIOLOGIST`, `CLINIC`, `EXAMINATION`, and the ternary directed relationship `Diagnosis`

of the destination macroentity that each element of the source macroentity can reach. Therefore, for each source node the cardinality corresponds to the number of instances that are allowed to participate in the association.

Figure 2.4 shows an example of a ternary directed relationship: macroentity `RADIOLOGIST` can be reached through the directed relationship `Diagnosis` from both `CLINIC` and `EXAMINATION`. Such a directed relationship is derived by the concept that a given radiologist performs a given examination in a given clinic. In particular, for what concerns with navigation paths, from a given clinic it is possible to navigate to radiologists who practice a given examination in such a clinic; also, from a given examination it is possible to navigate to radiologists who perform such an examination in a given clinic. Let us examine now cardinalities: we assume that each clinic have to provide at least one diagnostic service, and that each kind of examination could not be performed at all.

### Complex Directed Relationships

Whenever a conceptual association origins several directed relationships, a unique construct, a *complex directed relationships*, can indicate all navigation

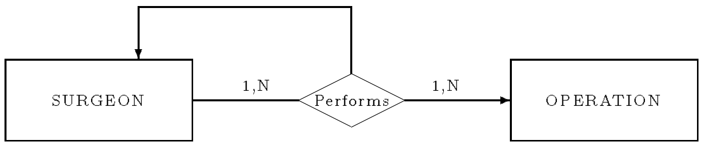



Figure 2.5: A complex directed relationship

paths derived from such an association.

We have seen that, in binary directed relationships, a binary association can give rise to a *symmetric* directed relationships, which indicates that navigation between the two nodes can proceed in both ways, being one the inverse of the other.

In n-ary directed relationships the same association can generate several directed relationships. For example, consider Figure 2.5: the ternary directed relationship `Performs` indicates that a given surgeon operates a given operation in team with other surgeons. In particular, given a surgeon one can navigate to each of the other surgeons who performs with him/her a given operation; moreover, given an operation, one can navigate to each surgeon who operates such an operation. Finally, `Performs` also expresses that, given a surgeon, it is possible to navigate to each of the operations such a surgeon does.

## 2.3 Attributes

Similarly to ER, attributes describe elementary properties of macroentities or directed relationships, and carry all the extensional information. Since a macroentity may involve multiple concepts, it is essential to specify for each of its attributes, whether it is *simple* (atomic) or *complex* (structured), and its *cardinality*, that is whether it is mono-valued or multi-valued [24, 14]. In the graphical representation, whenever minimum or maximum cardinality differs from one, it is explicitly indicated (see attribute `Topic`, which is multi-valued in Figure 2.6).

Let us consider now some examples. Figure 2.6 completes the NCM scheme presented in Figure 2.1: the attributes of macroentity `PHYSICIAN` and the attributes of macroentity `RESEARCH-GROUP` are all simple and mono-valued except for *(i)* the multi-valued attribute `Topic` of `RESEARCH-GROUP`, which represents information about major topics of research of the group, and *(ii)* the complex attribute `Name` of `PHYSICIAN`, which is composed by attributes `FName` and `SName`.

In Figure 2.7, the directed relationship `Location` connects macroentities `ANATOMIC-REGION`, which describes the main regions of human body, with `ORGAN`,

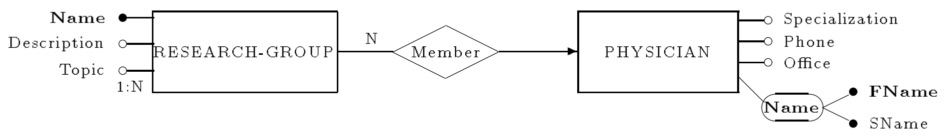



Figure 2.6: Attributes for macroentities `PHYSICIAN` and `RESEARCH-GROUP`

which explains functions of the of organs. Attribute `Position` specifies features of the association (for example, the heart is an organ whose location is in the left side of abdominal region).

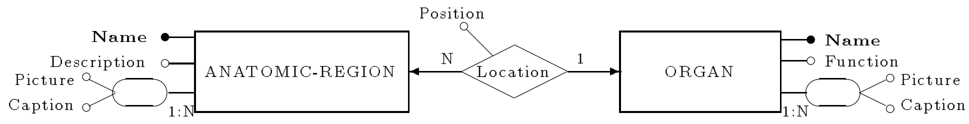



Figure 2.7: Attribute `Position` specifies features of the `Location` directed relationship

## 2.4 Descriptive Keys

With respect to identification, in NCM we have the notion of *descriptive key*. For each macroentity, a descriptive key is a subset of its attributes with two properties: *(i)* to be a super-key (in the usual sense) for the instances of the macroentity, and *(ii)* to be explicative about the corresponding instance, i.e. the user should directly infer the meaning of its values.

For example, consider the macroentity `PUBLICATION` in Figure 2.8: a descriptive key is made of attributes `Reference` and `Title`; although the reference alone would suffice to identify a publication, it does not convey enough meaning about the corresponding publication itself, so that also the title of the publication is needed to satisfy the second property. Note that, graphically, attributes that give rise to a descriptive keys are marked in boldface.

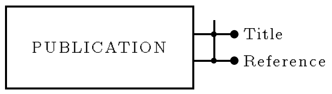



Figure 2.8: Descriptive key for macroentity `PUBLICATION`

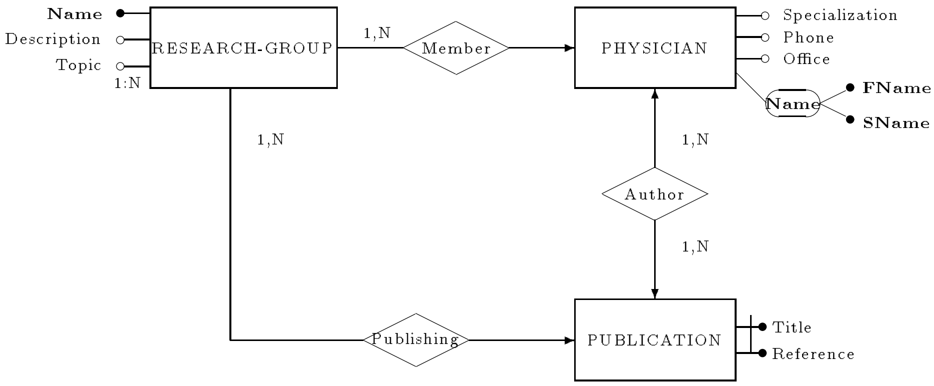



Figure 2.9: Redundant directed relationships in NCM schemes

## 2.5 Remarks on Macroentities and Directed Relationships

Although the role of macroentities and directed relationships is different from that of ER entities and relationships, the constructs are rather similar with their ER counterparts. Original features of the NCM constructs are the direction of traversal for directed relationships, and the notion of descriptive key for macroentities. Nevertheless, as we argued, macroentities aim at describing classes of objects whose instances are the atomic pieces of information users access, and the purpose of directed relationships is to define navigation paths to browse them. Thus, structured and multi-valued attributes assume a substantial and significant role for describing macroentities, and redundant directed relationships might often occur to provide useful navigation paths.

Consider for example Figure 2.9: in an ER perspective, the presence of the directed relationship `Publishing` would be redundant–as it can be obtained by means of relationships `Member` and `Author`, through the macroentity `PHYSICIAN`. Nevertheless, in the hypertext perspective, where paths correspond to connections that are directly available to the final user, it describes an important navigation from research groups to their publications.

## 2.6 Aggregations

Beside navigation paths based on conceptual associations between macroentities, an important and distinctive feature of hypertexts is the presence of an access structure. *Aggregation* is the NCM primitive to model the hypertext

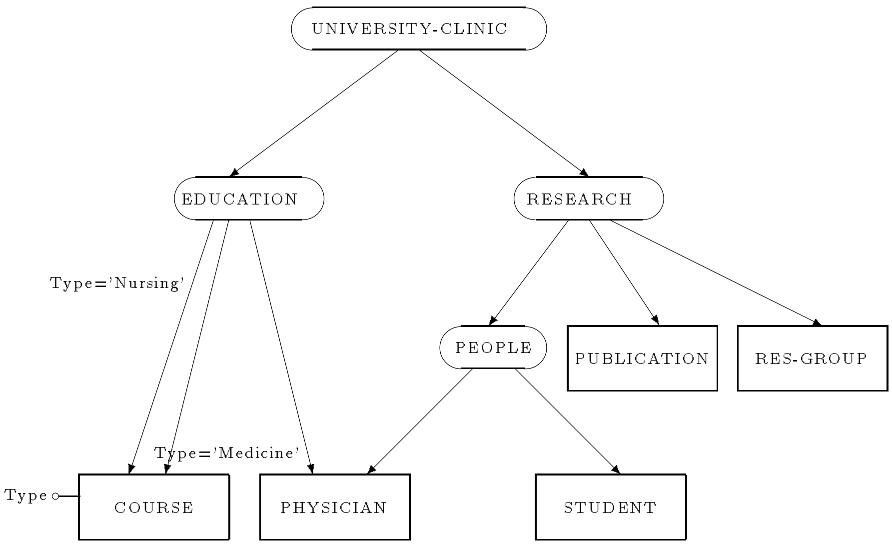



Figure 2.10: Aggregations `CLINIC`, `EDUCATION`, `RESEARCH`, and `PEOPLE`

access structure: an aggregation node is a means to reach the involved concepts (macroentities) or, in turn, other aggregations. In hypertext, the access structure is directly available by the final user, and it is used to aggregate macroentities on the basis of conceptual aggregations or classifications.

Graphically, we represent aggregation as rounded rectangles, and arrows are used to indicate the nodes that participate the aggregation, and that are accessible from the aggregation.

Let us discuss aggregations by means of an example: Figure 2.10 shows a portion of a NCM scheme dealing with pieces of information concerning the academic activities (education and research) of a university clinic. The node `CLINIC` is an example of an aggregation, which models the main entry point of the information system, and leads to other aggregation nodes (essentially acts as a menu): `EDUCATION` and `RESEARCH`. The former conducts to macroentities `PHYSICIAN` and `COURSE`; the latter to macroentities `PUBLICATION` and `RESEARCH-GROUP`, and to the aggregation `PEOPLE`, which in turn leads to the macroentities `PHYSICIAN` and `STUDENT`.

Sometimes the participation of a macroentity to an aggregation is only partial, in the sense that only a subset of the instances of a macroentity is involved. This is modeled in NCM by labelling aggregation links: each label is associated with a predicate on instances of the destination node, and is used to specify that only instances that satisfy the predicate are considered as part of the aggregation.

In the example discussed above, it could be reasonable to distinguish the

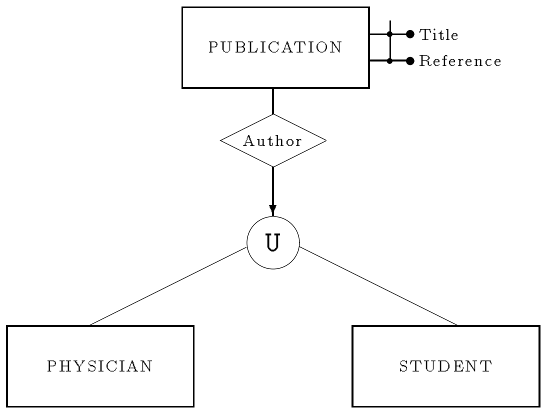



Figure 2.11: Union node connects publication to either physicians or students

access to nursing and medicine courses: in Figure 2.10 two links with different labels ("Type=medicine" and "Type=nursing") are used to this end.

## 2.7 Union Nodes

In order to model effective navigation paths, a directed relationship may involve the *disjoint union* of several macroentities as destination node. For example, assume that both physicians and students can author publications. Suppose we are interested in modeling the navigation path that starts from `PUBLICATION` and leads to the corresponding authors. We do not aim at introducing an "abstract" type that would describe the class of authors (generalizing the classes of physicians and students), but we are motivated by the pragmatic purpose of defining a navigation path that drives to each of the authors of a given publication, being it either an instance of `PHYSICIAN` or an instance of `STUDENT`.

A *union node* satisfies this end: it allows us to model, as destination node of a directed relationship, the disjoint union of several existing macroentities.

In the graphical representation a circled "U" symbolizes union nodes. Figure 2.11 illustrates the example discussed above: the disjoint union of physicians and students is the destination of directed relationship `Author`, whose source is the macroentity `PUBLICATION`.

Union nodes hide the peculiar nature of the destination instance from the starting macroentity. In fact, in the previous example, each navigation from an instance of `PUBLICATION` can lead either to a student, or to a physician. Figure 2.12 shows an example of the extension of the NCM scheme in Figure 2.11: circles symbolize instances of macroentity, and arrows correspond

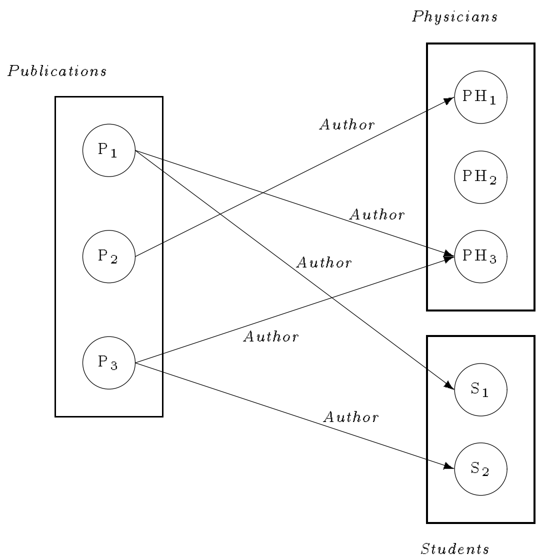



Figure 2.12: Extension of the NCM scheme in Figure 2.11: `Author` links publications to both students and physicians

to instances of directed relationships (that is hyper-links). At the instance level, the directed relationship `Author` has both instances (hyper-links) that connect publications to students, and instances that connect publications to physicians.

On the contrary, consider Figure 2.13, which describes navigations from research groups to their members. Both physicians and students can join (at most) one research group, but differently from the previous example, the designer emphases, in the source macroentity, navigation paths that lead to members who are physicians, and navigation paths to members who are students. To this end, two distinct directed relationships occur (one for each destination macroentity): the former, `PMember`, describes the navigation from a research group to its physician members, the latter, `SMemeber` concerns with the navigation from a research group to its student members. Figure 2.14 shows the extension of the NCM scheme in Figure 2.13: note that, in this case, links to physicians and links to students have different types, as denoted by the labels, because they refer to different relationships.

## 2.8 Roles

A hypertext can model rather complex realities which pertain to several aspects. For example, we have seen that a hypertext dealing with the presen-

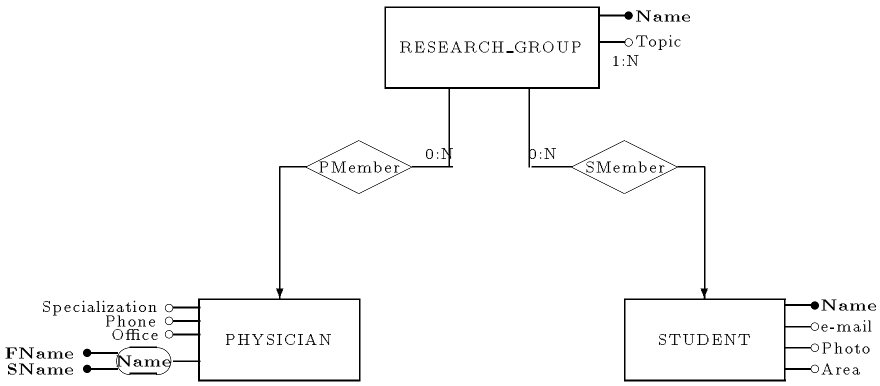



Figure 2.13: Two directed relationship model distinct navigation paths from research groups

tation of a university clinic may embrace both pieces of information about research, and descriptions of the educational activities.

Usually, the hierarchical organization of aggregations allows the hypertext to achieve an overall structure of navigation that separates the different facets of the application domain. In the example of Figure 2.10, starting from the home page, distinct navigation paths drive to macroentities that model classes of objects belonging to either the educational or the research context.

Nevertheless, there exist some macroentities that play active roles in various contexts of the application domain. For example, physicians participate both in the research and educational activities, as researchers and professors, respectively. Such macroentities have a heterogeneous nature, and each of their attributes and directed relationships usually concerns with just one specific aspect of the modeled reality. In the physicians example, a directed relationship to courses specifically deals with the teacher role.

Sometimes, it may be useful to present the same instance of a macroentity in distinct nodes of the hypertext, each node emphasizing a specific aspect of the domain concept by means of an appropriate subset of attributes and directed relationships.

NCM provides the *role* construct, which is inherited from object oriented data models [40, 10, 31], in order to allow designers to model the various roles a macroentity can play. Defining roles corresponds to split the set of attributes and directed relationships of a given macroentity in several (possibly overlapped) partitions, each partition corresponding to a particular presentation of the macroentity. Roles act on the extension of the macroentity: they do not create new classes of objects, but each instance of a macroentity is composed by several parts, described by roles.

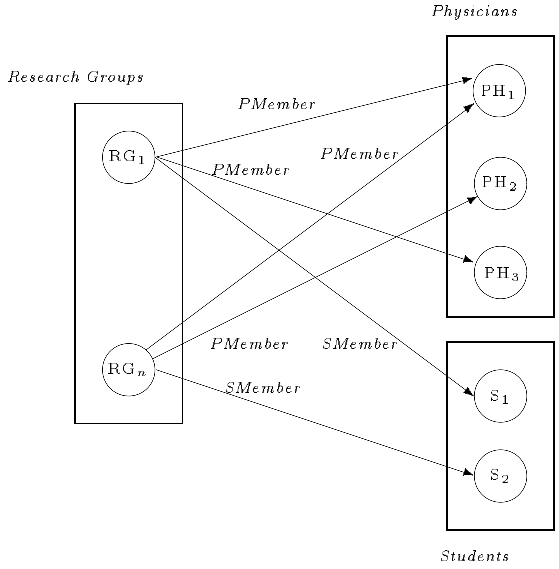



Figure 2.14: Extension of the NCM scheme in Figure 2.13: member links to physicians and links to students are distinct

For example, consider the class of physicians: by means of roles we can represent the various aspects of each instance: one role describes properties related with teaching, a different one models features concerning with the research activity.

Different roles of the same macroentity can share attributes and directed relationships that describe general properties and associations that do not depend on any specific role. For example, name, surname and specialization of physicians arise in both their teacher and researcher oriented descriptions. It should be noted that directed relationships that roles can share have to be exiting from the macroentity, since entering relationships must lead to only one node. We introduce the notion of *default role*: it is the role that corresponds the node where an entering directed relationship usually leads to, if it is not otherwise specified.

In order to model navigation paths between role partitions of the same macroentity, NCM has *role-links*: each role-link represents a one-to-one navigation-path that involves a pair of roles of the same macroentity. Like directed relationships, symmetric role-links may occur.

Graphically, each role is drawn like a circle, which is labelled with the role name, and is connected by an edge with the macroentity it refers to; the role-specific attributes are defined on the role symbol, while shared attributes (and shared directed relationships) are drawn on the macroentity; a double circle

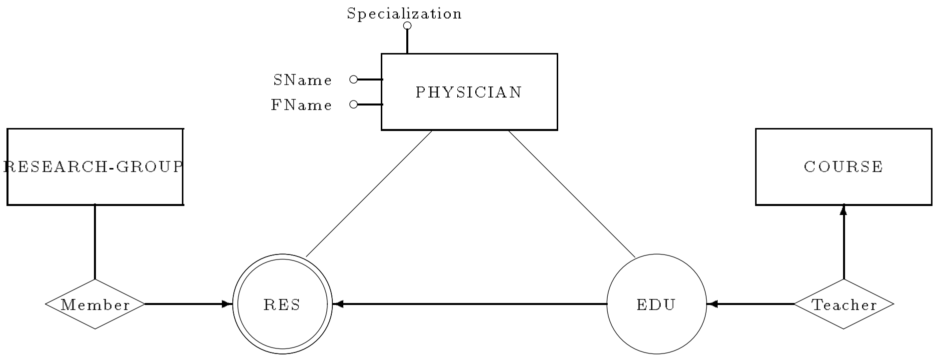



Figure 2.15: Roles partitioning macroentity `PHYSICIAN`

marks the default node; arrows connecting role nodes express role-links.

Figure 2.15 shows our graphical conventions, and illustrates the introduced concepts: to model the twofold activity of physicians, two roles are associated to the macroentity `PHYSICIAN`. The former, which we label `RES`, describes the researcher topics, the latter, `EDU`, contains information of physician as teacher. Both of the research and education oriented partitions need attributes of general interest, such as name, specialization degree, room and phone number. Other attributes, which concern with the specific role, are associated to either the educational or the research partition. Also, since each role may have its own directed relationships, we can connect it to the pertinent context. For instance, we connect the researcher role to macroentities `RESEARCH-GROUP` and `PUBLICATION` (the latter not shown in Figure 2.15) by means of directed relationships `Member` and `Author`, respectively, and the educational role to macroentity `COURSE` by means of the directed relationship `Teacher`. A role-link connects the research-oriented partition to the teacher-oriented one. Finally, the researcher role is elected as the default role, thus all directed relationships that have `PHYSICIAN` as destination node leads to the role partition. Figure 2.16 depicts a sample of the extension of the NCM scheme in Figure 2.15. Note that each instance of the macroentity `PHYSICIAN` corresponds with two node (one for each role partition), which are connected by the instances of the role-link. Also, note that links to courses starts from the educational role instances, whereas links with research groups involve instances describing the research oriented role.

Figure 2.17 summarizes the graphic representation of NCM constructs.

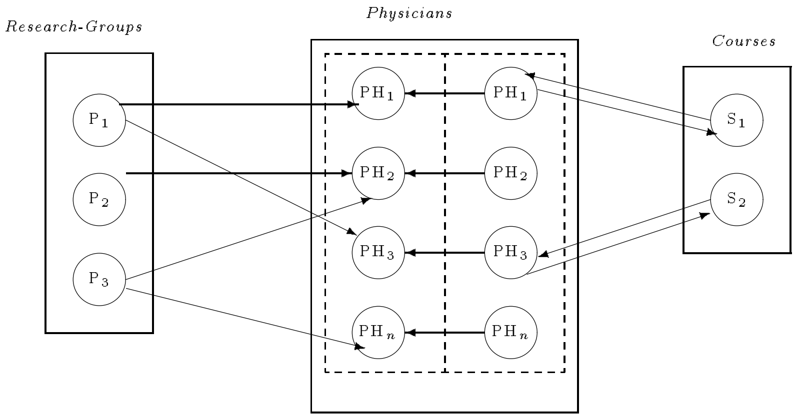



Figure 2.16: The extension of NCM scheme in Figure 2.15

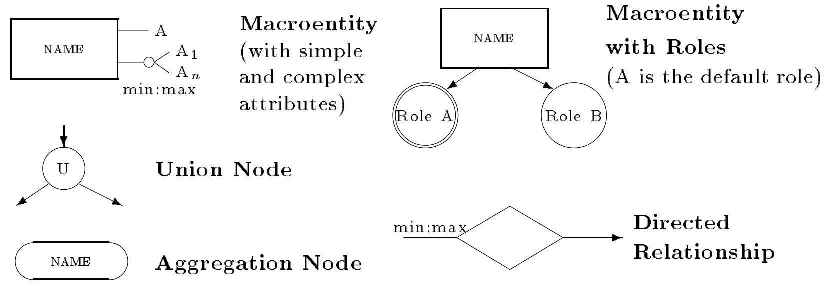



Figure 2.17: Graphical Representation of NCM Constructs

# Chapter 3

# ADM: a Logical Data Model for Hypertexts

Coherently with their conceptual nature, NCM schemes illustrate how concepts can be navigated in the target hypertext. Nevertheless, actual Web hypertexts are essentially graphs of pages: these two ways of abstracting the organization on information can be rather far away from each other.

An hypertext logical data model provides constructs that allow the designer to describe in a tight and concise way the structure of HTML pages by abstracting their logical features. We propose to this end a specific data model, called the ARANEUS *data model (*ADM*)* [11]; we say that such a model is *page oriented*, in the sense that it recognizes the central role that pages play in this framework.

In the following we present the constructs of ADM and discuss them by means of several examples inspired by the Web interface of the MrBrAQue system, and by the OncoLink bibliographic server at the University of Pennsylvania [3]. [1]

## 3.1 ADM Page Schemes

The fundamental feature of ADM is the notion of *page scheme*, which resembles the notion of relation scheme in relational databases or class in object-oriented databases: a page scheme is an intensional description of a set of Web pages with common features. An instance of a page scheme is a Web page, which is considered as an object with an identifier (the URL) and a set of attributes.

[1] Although in this context ADM is addressed as a tool to model new Web sites, we also experienced it in order to describe existing Web sites (for querying purposes).

**Unique Page Scheme** There is one specific aspect in this framework with no counterpart in traditional data models. On a page scheme a special constraint can be specified in order to model an important case in this context: when a page scheme is "unique", it has just one instance, in the sense that there are no other pages with the same structure. Typically, at least the home page of each site falls in this category.

In ADM the content of Web pages is described by means of *attributes*, which may have simple or complex type.

### 3.1.1 Simple Attributes

*Simple attributes* are mono-valued and correspond to atomic pieces of information, such as *text*, *images* or other multimedia types. *Links* are simple attributes of a special kind, used to model hypertextual links; each link is a pair *(anchor, reference)*, where the *reference* is the URL of the destination page, possibly concatenated to an *offset*, inside the target page scheme, and *anchor* is a text or an image. Anchors for links may either be constant strings, or correspond to tuples of attributes. An offset, whenever needed, must fit an attribute of the destination page.

Consider for example the Web page in Figure 3.1, which presents data about an examination in the MrBrAQue Web interface. We can see that there is a set of elements that appear in this page: the date of the examination, the type of examination, a link to the patient's personal data, whose anchor is the name of the patient himself, the name of the physician who is responsible for the patient, the name of the radiologist who performs the examination, and a link to the actual report of the examination. It is natural to model the structure of these pages as a page scheme, with several attributes, as shown in Figure 3.2. Also, it should be pointed out that this abstract description fits for all pages ( in the Web interface of the MrBrAQue system) with the same structure, i.e. all those pages which represent the front page of an examination for a different patient or for a different date.

### 3.1.2 Complex Attributes

*Complex attributes* are multi-valued and represent (ordered) collections of objects, that is, *lists* of tuples. Component types in lists can be in turn multi-valued, and therefore nested lists are allowed. It should be noted that we have chosen lists as the only multi-valued type since repeated patterns in Web pages are physically ordered.

Figure 3.3 shows a page from OncoLink [3]. Such a page has one simple attributes, that is a class of disease, but it also has a *complex* attribute, that is a list of diseases. This is a multi-valued attribute of the page, and can be

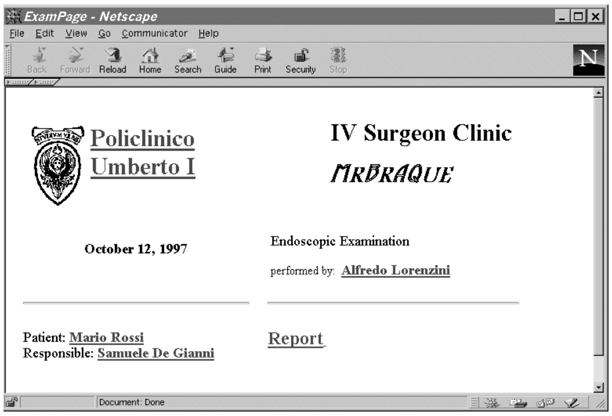



Figure 3.1: A page from the MrBRAQue Web Interface

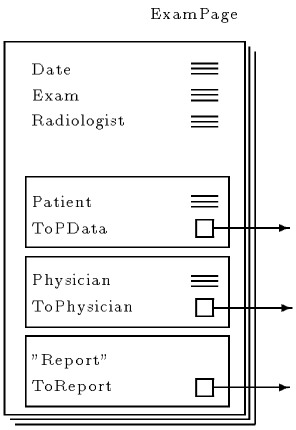



Figure 3.2: Page-scheme ExamPage

modelled as a list, whose elements are links: anchor of each link is the name of a disease, and the reference points to a page containing further pieces of information about such a disease. Figure 3.4 shows (graphically) how ADM constructs model such a page.

Attributes may be labelled as "optional" in order to allow null values.

### 3.1.3 Forms

An important construct in Web pages is represented by *forms*. Conversely, forms are used to execute programs on the server and dynamically generate pages. ADM provides a *form type*: in order to abstract the logical features of an HTML form, we see it as a *virtual list of tuples*; each tuple has as many

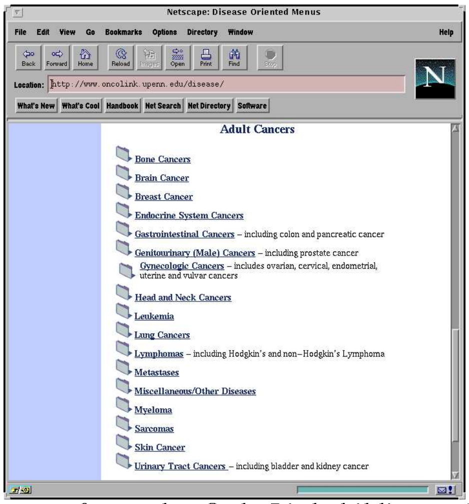



Figure 3.3: An actual page from the OnkoLink bibliographic service at the University of Pennsylvania

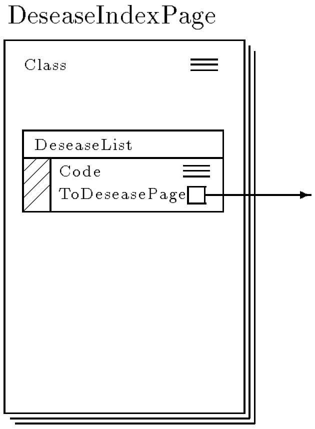



Figure 3.4: ADM Page-scheme for page in Figure 3.3

attributes as the fill-in fields of the form, plus a link to the resulting page;[2] such lists are *virtual* since tuples are not stored in the page but are generated in response to the submission of the form.

### 3.1.4 Heterogeneous unions

ADM provides a *heterogeneous union* type, in order to provide flexibility in modelling, according to the heterogeneous nature of the Web.

Consider again the Web interface of MrBraQue: in order to allow for an

[2] Forms introduce several specific data types, such as *check-boxes*, *radios* or *selections*. We ignore these aspects here for simplicity, and consider only attributes of type text. All ideas can be easily generalized to the most general case.

effective access to information about a specific patient, the system provides a page that contains a form: by specifying a string corresponding to the name of a patient, personal data about such a patient are returned. Figure 3.5 shows the actual page with the discussed form. The form can be seen as a virtual list of tuples with a link attribute: the anchor of the link is the text entered as a string to search, the reference can be considered as a link to the page generated by the corresponding search.

Let now consider the behavior of the search form: when a string is specified, the patient database is searched; if a single name matching the query string is found, the patient's page, with her/his personal information, is returned; otherwise, if the query string matches several names, a different page is returned, which contains a list of links to the matching patients. By means of union we can easily model this involved behavior, as shown in Figure 3.6.

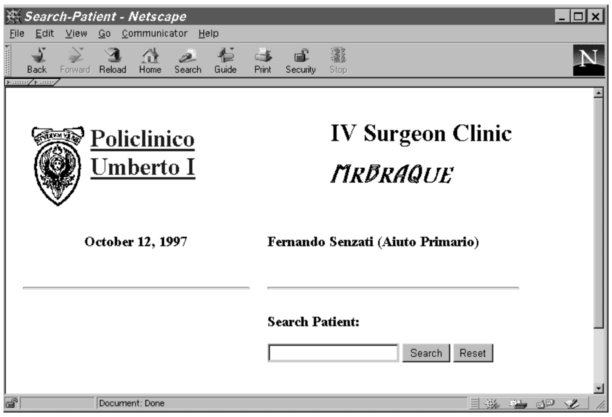



Figure 3.5: The form for searching data about patients in the MrBraQue Web interface

## 3.2 Constraints

The hypertextual organization of information induces a high degree of redundancy, which appears in Web pages in two ways.

First, many pieces of information are replicated over several pages. Consider for example Figure 3.6: the value of the anchor attribute `Patient` in page scheme `PatientListPage` must equal the value of the `Patient` attribute in the destination page scheme `PatientPage`. The reason of this kind of redundancy can be associated to the nature of the anchor component of link attributes, because it is commonplace that the anchor of a link corresponds to the value of an attribute of the destination page. A special case of this kind of redundancy occurs when, following a link from a source page scheme, the

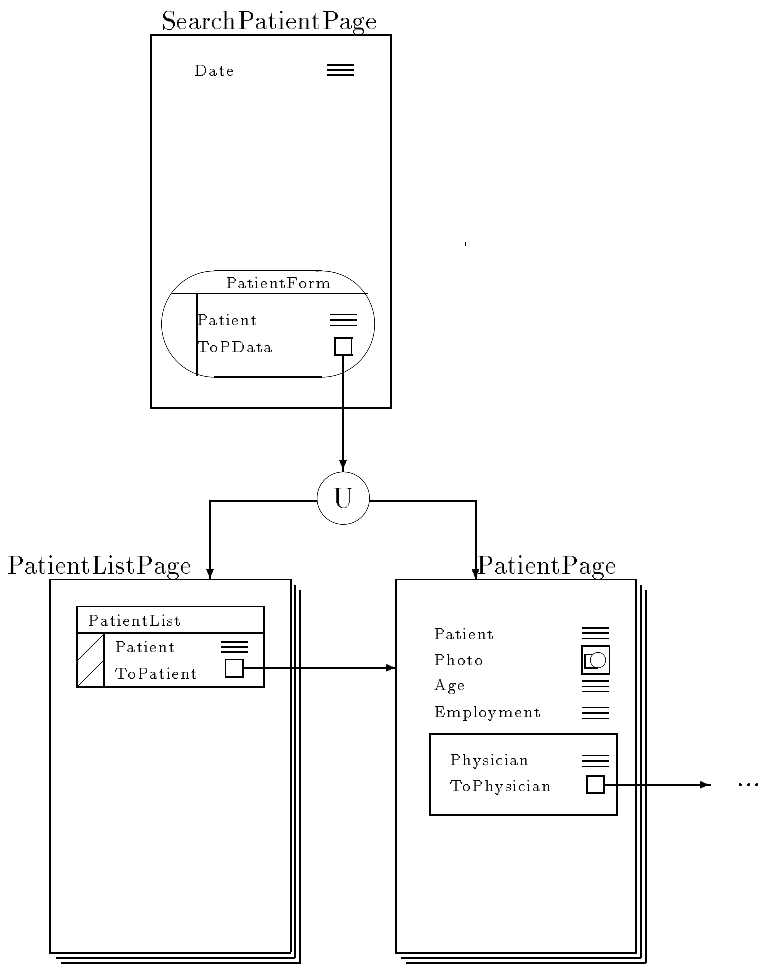



Figure 3.6: Forms and Heterogeneous Unions in ADM

value of an attribute in the target page scheme is a constant value for every instance. Moreover, replications of pieces of information over several pages often arise for the sake of clarity. For example, in an hypertext providing information about patients in an hospital information system, we can expect that all pages concerning data about a certain patient present (at least) the name and the age of the patient.

Second, redundancy emerges also in the navigation paths of the hypertext. In fact pages can be usually reached following different navigation paths in the site. For example, the home page of a physician working in a university clinic could be reached both from pages concerning the taught courses and from pages presenting the personnel.

In order to capture these redundancies we enrich the model with two kinds of integrity constraints, *link constraints*, and *inclusion constraints*.

### 3.2.1 Link Constraints

A *link constraint* [38] is a predicate associated with a link. It is used to document the fact that the value of some attribute in the source page scheme equals the value of another attribute in the target page scheme. For example, with respect to the previous example (see Figure 3.6), this is the case for attributes `Patient` and `Name` in page-schemes `PatientListPage` and `PatientPage` respectively: we expect that, whenever the link `ToPatient` is followed, the value of attribute `Patient` in page-scheme `PatientListPage` equals the value of attribute `Name` in page-scheme `PatientPage`. The following predicate is used to documented such a constraint:

```
PatientListPage.Patient = PatientPage.Patient
```

Because of the nested structure of page-schemes, a link constraint can also document that the value of some attribute in the source page equals one of the values of a nested attribute in the target page-scheme. Consider for example Figure 3.7: page-scheme describes (`PhysicianPage`), and (`PublicationPage`) contains a list (`PaperList`) of publication: each item of the list, beside some simple attributes describing title and reference of each paper, contains in turn a list, `AuthorList`, which provides the names of all authors. If we suppose that each instance of the link from `ToPublication` leads to an instance of `PublicationPage` that lists only the papers of the source physician, then we expect that the name of the source physician appears in all the instances of the list of author. The following predicate documents such a constraint:

```
PhysicianPage.Name ∈ PublicationPage.PaperList.AuthorList
```

To formalize, given two page-schemes, $P_1$ and $P_2$ connected by a link $ToP_2$, a *link constraint* between $P_1$ and $P_2$ is any expression of the following forms:

- $A = B$ or $A = v$, where $A$ is a mono-valued attribute of $P_1$, $B$ a mono-valued attribute of $P_2$, and $v$ is a constant;

- $A \in B$ or $v \in v$, where $A$ is a mono-valued attribute of $P_1$, $B$ a multi-valued attribute of $P_2$, and $v$ is a constant.

Given an instance of the two page-schemes, we say that the link constraint holds if: for each pair of attribute $t_1 \in P_1, t_2 \in P_2$, then attribute $ToP_2$ of $t_1$ equals attribute URL of $t_2$ if and only if attribute $A$ of $t_1$ equals (belongs to) attribute $B$ of $t_2$.

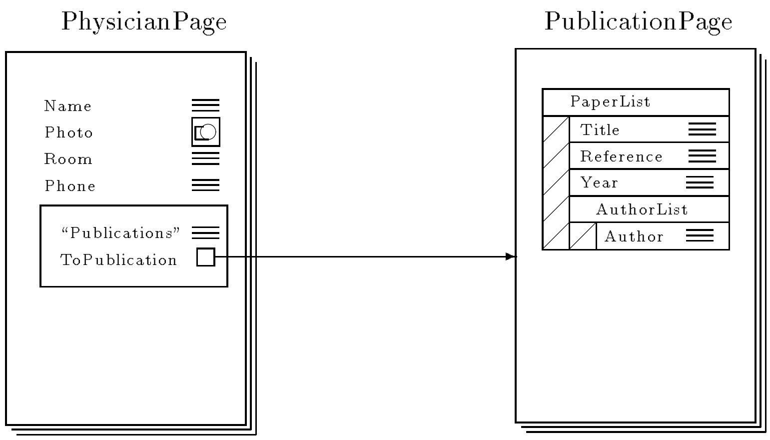



Figure 3.7: Page-schemes `PhysicianPage` and `PublicationPage`

### Implicit Link Constraints

Due to the very nature of links, a number of link constraints involve URLs. In fact, for each link attribute in a page, the value of a link attribute always must equal the URL of the target page. As an example, consider again attribute `ToPatientPage` in page-scheme `PatientListPage`: for any of the corresponding pages, it must be the case that: `PatientListPage.ToPatientPage = PatientPage.URL`. These constraints are *implicit* in the notion of link reference, and do not represent redundancies, so we ignore them.

## 3.2.2 Inclusion Constraints

Besides link constraints, we also extend in our context the notion of *inclusion constraint* [7], in order to reason about containment among different navigation paths. Consider Figure 3.8: it can be seen that page-scheme `PhysicianPage` can be reached either from `PersonnelPage` or from `PatientPage`. Since page-scheme `PersonnelPage` corresponds to the list of all physicians, it is easy to see that the following inclusion constraints hold:

$$\texttt{PatientPage.ToPhysician} \subseteq \texttt{PersonnelPage.PhysList.ToPhysician}$$

$$\texttt{PatientPage.Physician} \subseteq \texttt{PersonnelPage.PhysList.Physician}$$

Note that the inverse containments do not hold in general. For example, following the path that goes through patient pages, only physicians that are responsible for at least one patient can be reached; but there may be physicians who cannot be responsible for any patients.

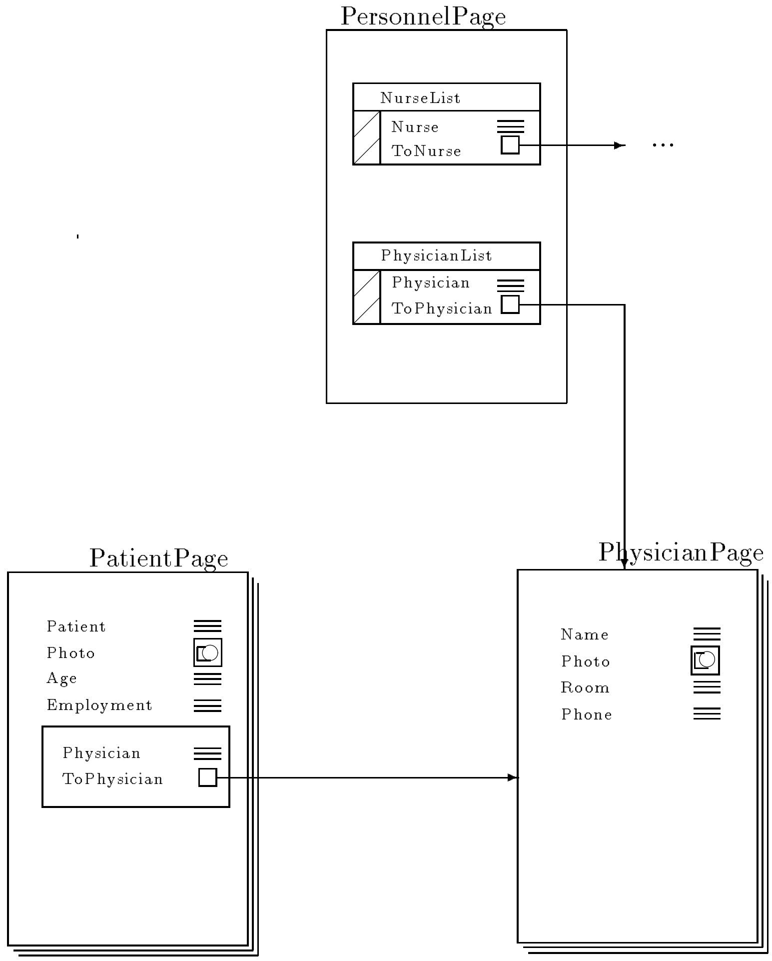



Figure 3.8: Inclusion Constraints and Link Constraints

To formalize, given a page-scheme $P$, and two attributes $A_1, A_2$ towards $P$ in $A_1$ and $A_2$, an *inclusion constraint* is an expression of the form: $P_1.A_1 \subseteq P_2.A_2$. Given instances $p_1$ and $p_2$ of each page-scheme, we say that the inclusion constraint holds if: for each tuple $t_1 \in p_1$, there is a tuple $t_2 \in p_2$, such that the value of $A_1$ in $t_1$ equals the value of $A_2$ in $t_2$. As usual, two constraints of the form $P_1.A_1 \subseteq P_2.A_2$, $P_2.A_2 \subseteq P_1.A_1$ may be written in compact form as $P_1.A_1 \equiv P_2.A_2$.

## 3.3 ADM Scheme

An ADM *scheme* is made of $(i)$ a set of page-schemes (connected by links), and $(ii)$ a set of link and inclusion constraints.

To formalize, we can define ADM types as follows. Given a set of *base types* containing the types `TEXT` and `IMAGE`, a set of *attributes* and a set of *page-scheme names*, the set of ADM *types* is recursively defined as follows (each type is either monovalued or multivalued):

- each base type is a *monovalued* ADM *type*;
- LINK(REFERENCE$D$, ANCHOR$(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$) is a *monovalued* ADM *type*; $D$, the destination of the link, is *(i)* either a page-scheme name, $P$, *(ii)* or a *union type*, $P_1$ UNION $P_2$ UNION $\ldots P_n$, where each $P_i$ is a page-scheme name; $A_1, A_2, \ldots, A_n$ are attributes and $T_1, T_2, \ldots, T_n$ are base types;
- LIST OF$(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$ is a *multivalued* ADM *type*, if $A_1, A_2, \ldots, A_n$ are attributes and $T_1, T_2, \ldots, T_n$ are ADM types;
- FORM(LINK(REFERENCE$D$, ANCHOR$(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$)) is a *multivalued* ADM *type*, if $A_1, A_2, \ldots, A_n$ are attributes and $T_1, T_2, \ldots, T_n$ are ADM types; $D$ denotes the name of the page-scheme generated in response to the submission of the form.

An ADM *page-scheme* has the form $P(A_1 : T_1, A_2 : T_2, \ldots, A_n : T_n)$, where $P$ is a page name, each $A_i$ is an attribute and each $T_i$ is an ADM type. Attributes may be labeled *optional*. The page-scheme may be labeled as *unique*.

The page schemes discussed in the previous examples can be defined as follows using the ARANEUS *data definition language (DDL)*, whose syntax naturally follows from the previous definition. Examination pages (Figures 3.1 and 3.2) have the following scheme, in which only monovalued attributes appear:

```
PAGE SCHEME ExamPage:
               Date:       TEXT;
               Exam:       TEXT;
               Radiologist:TEXT;
               ToPatient:  LINK (REFERENCE PatientPage,
                                 ANCHOR (Patient:  TEXT));
               ToPhysician:LINK (REFERENCE PhysicianPage,
                                 ANCHOR (Physician:  TEXT));
               ToReport:   LINK (REFERENCE ReportPage,
                                 ANCHOR ("Physician"));
END PAGE SCHEME
```

Pages for diseases (Figure 3.4) have a multivalued attribute, i.e. the list of diseases, and can be described as follows:

```
PAGE SCHEME DiseaseIndexPage:
            Class:        TEXT;
            DiseaseList:  LIST OF (ToDiseasePage:
                                   LINK (REFERENCE DeseasePage,
                                         ANCHOR (Code :  TEXT)));
END PAGE SCHEME
```

Finally, the unique page containing the form to search patients (Figure 3.6) has the following structure:

```
PAGE SCHEME SearchPatientPage UNIQUE:
              PatienForm: FORM (REFERENCE:
                                   PatientListPage UNION PatientPage,
                                   ANCHOR (Patient:  TEXT));
END PAGE SCHEME
```

Graphically, an ADM scheme results in a directed multi-graph; nodes in the scheme graph are page schemes; the symbol for a unique page scheme recalls a single page, whereas that for non-unique page schemes recalls a "stack" of pages; edges are used to denote links.

Figure 3.9 summarizes the graphical representation of the ADM constructs. Redundant attributes in the site are documented using link-constraints. Containment between different navigation paths to the same page-scheme are documented by inclusion constraints.

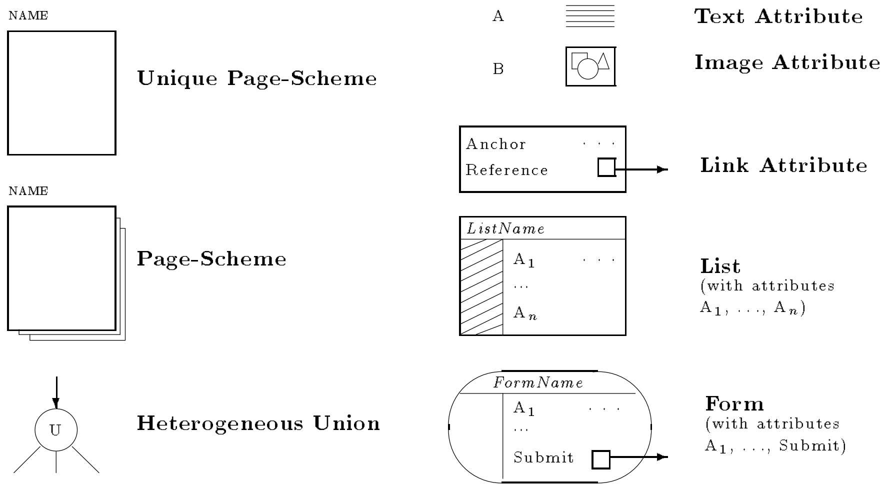



Figure 3.9: Graphical representation of ADM constructs

# Part II

# Hypertext Design

As we discussed in Chapter 1, our approach for designing a data-intensive Web hypertext separates the overall design activity into two well distinguished and yet tightly interconnected processes: the *database design* process and the *hypertext design* process.

The objective of hypertext design process is twofold. On the one hand, the structure of the hypertext should be designed according to the main concepts of the application domain and providing navigation paths that allow users to browse information in the hypertext. On the other hand, the organization of data inside Web pages should be carefully described. In the spirit of databases [14], we distinguish a *conceptual* and *logical* design phase, in order to isolate system features from higher-level perspectives, and facilitates further maintenance of the hypertext. Hence, the hypertext design process consists in two phases: *Navigation Conceptual Design*, and *Hypertext Logical Design*.

Navigation conceptual design aims at providing a hypertextual description of the application domain: the main classes of objects to be published trough by the Web hypertext are identified and described, and navigation paths between them are designed. Navigation conceptual design starts from a conceptual description of the application domain (usually an ER scheme), and produces a NCM scheme, which provides a conceptual description of the hypertext. Chapter 4 deals with this process.

The goal of hypertext logical design is to describe the logical organization of information in Web pages. Hypertext Logical Design starts from a NCM scheme and results in an ADM scheme. An ADM scheme consists in an abstract description of the logical organization of data in pages: an ADM scheme is a graph where nodes represent pages, and edges links between pages; attributes describe the content of pages. Hypertext Logical Design is discussed in Chapter 5.

# Chapter 4

# Navigation Conceptual Design

The goal of navigation conceptual design is to build a navigation conceptual scheme (according to the NCM model), starting from a conceptual description of the underlying application domain. We assume that an ER scheme is the vehicle that models the reality of interest: although the design process could be easily extended to other formalisms, we refer to the ER data model [21] because of its expressiveness, and because it is rather standard in database design. In fact, as we discussed in Chapter 1, the overall organization of our methodology is modular, and can adapt even to produce an hypertext starting from an existing database.

Navigation conceptual design typically involves three main activities: *(i)* deciding which classes of objects from the application domain will be part of the hypertext; *(ii)* choosing the paths to navigate between them; *(iii)* designing the hypertext access structure.

The methodology we present to design the NCM scheme consists of two main successive processes. First, we generate a skeleton of the NCM scheme from the ER description: we introduce ER-NCM *translation primitives* to describe and document this process that maps ER schemes into NCM constructs; therefore, we refine the resulting NCM scheme: NCM *refinement primitives* allows for a progressive improvement of the hypertextual organization of information.

The chapter is organized as follows. In Sections 4.1 and 4.2 we discuss general issues for generating an NCM scheme starting from an ER description, and present ER-NCM translation primitives. Section 4.3 deals with NCM refinement primitives.

## 4.1 Mapping ER schemes into NCM

A conceptual scheme of the navigation has two main objective: *(i)* providing a representation that reflects concepts and relationships of the application domain, and *(ii)* offering navigation paths that allow for a coherent and suitable browsing of information.

In our process of hypertext design, an ER scheme plays the role of input product, providing an accurate and sound description of the application domain. Thus, conceptual design of navigation essentially consists in mapping an ER scheme into an NCM description.

In Chapter 2, we have seen that NCM, our formalism to describe a conceptual scheme of navigation, tailors and extends constructs of the ER data model in the hypertext framework. The mapping process takes into account analogies and singularities between the constructs of the two models. On the one hand, primitives from the ER scheme naturally and progressively give rise to the NCM constructs that compose the conceptual scheme of the navigation. On the other hand, although some similarities occur between NCM and ER, there are some substantial differences, due to the very nature and purposes of NCM, that influence the mapping process. First, directed relationships express the direction of traversals, IS-A relationships are not present in NCM, whereas union nodes and roles are supported with specific modeling goals, and aggregations do not have any counterpart in the ER model. Moreover, besides differences in the basic constructs, there exist sources of divergence between the two data models that involve overall features of the schemes. In fact, conceptual schemes of data typically satisfy qualities that are in contrast with properties a navigation scheme should gain. For example, a conceptual scheme of navigation need a large amount of redundancy, which diverges with minimality of ER schemes, and organizes information in order to reach completeness in atomic constructs, which leads to violation of normal forms.

Hence, the mapping of ER schemes into NCM description cannot be thought as a simple step-by-step translation process, but it relies on several decisions by designers. In this section we present a general approach designers can apply in the mapping process; in particular we organize the process in three main phases, each one focusing on the design of a specific NCM construct. The first phase aims at giving rise to macroentities and roles; the second phase deals with the design of directed relationships and union nodes; finally, the third phase treats the creation of aggregations. We illustrate each phase in a separate subsection, and discuss how decisions designers take in each phase may influence subsequent ones.

### 4.1.1 Macroentity Design

Macroentity design deals with two main activities: (i) choosing the main concepts to be represented in the hypertext, and associating with a macroentity each of them; (ii) describing macroentities properties (attributes).

Macroentities are naturally derived from ER entities: each entity representing objects that are considered relevant and independent in the hypertext originates a macroentity. We indicate as "originating entity" each ER entity that leads to the creation of an NCM macroentity: we associate one, and only one generating entity, to each macroentity. Usually, each macroentity is given the same name as the originating entity; nevertheless, new names can be introduced, which better identify the underlying concepts.

As discussed in Chapter 2, NCM does not have IS-A relationships, whereas union nodes and roles are supported. In the design of macroentities, several decisions occur in analyzing IS-A relationships, since different strategies may arise depending on both designer preferences and features of the ER scheme. In 4.1.1 we address this issue.

Since in NCM macroentity instances are the atomic pieces of information that users access, each macroentity has to correspond to concepts whose instances have autonomous and significant existence in the hypertext. Therefore, it is usual that portions of the starting ER scheme participate in the mapping process, providing their attributes to the risen macroentity. In 4.1.1 we discuss the activity of mapping entities and relationships into NCM attributes.

We point out that the description of macroentities by merging, and the derivation of macroentities from IS-A hierarchies are not competitive, but rather complementary activities: designers can apply the merging process in order to complete the description of any of the macroentity derived from the starting ER scheme.

#### Deriving Macroentities from IS-A Hierarchies

Mapping IS-A hierarchies into NCM constructs is a complex activity, which can give rise to several alternatives, and suggest the introduction of roles as well. In the following, we introduce three main strategies for mapping entities participating IS-A hierarchies into NCM. Choosing one of the proposed approaches depends both on properties of the examined hierarchy, and on subjective decisions by designer, which concerns with the significance of the domain concepts the IS-A relationship models.

Each strategy influences further phases of design, favoring the introduction of different constructs (for example, roles vs. union nodes), and conditions the overall structure of the navigation in the resulting hypertext.

**Deriving Macroentities from Leaf Nodes** When the ER scheme contains an IS-A relationship, if the children (or leaves, in case of multilevel hierarchy) are relevant, then, each of them give rise to a macroentity. The significance of entities in an IS-A hierarchy depends on the estensional properties: if each leaf node is well characterized by its attribute set and its relationships, then it makes sense to generate autonomous hypertext elements, i.e. macroentities, by them. Clearly, each macroentity, beside the attributes coming from the originating entity, has the attributes from the parent entity of the IS-A relationship, because of the inheritance constraint of ER hierarchies.

Consider for example Figure 4.1: it illustrates an IS-A hierarchy that classifies people in a hospital. Now, entities `PHYSICIAN` and `NURSE` correspond to concepts whose instances may have an autonomous existence in the hypertext; on the contrary, the generalization entity `CARING-PERSONNEL` is not relevant, that is, we do not expect to find any instance of it in the hypertext. Hence, macroentities `PHYSICIAN` and `NURSE` are generated from the corresponding entities. As it will be discussed in 4.1.2 and 4.1.3 the parent entity may further participate in creating an aggregation or a union node.

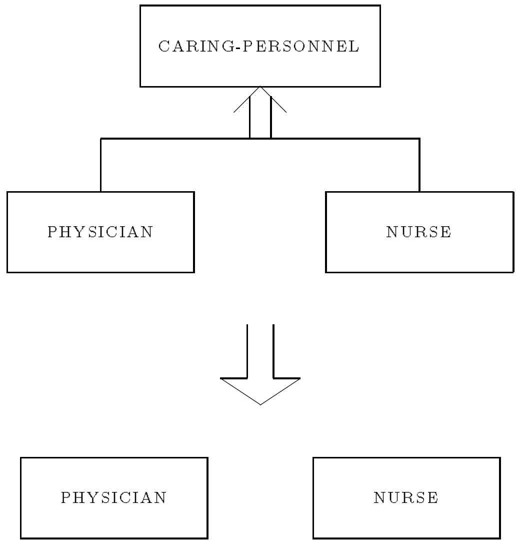



Figure 4.1: Deriving Macroentities from leave nodes of IS-A hierarchies

**Deriving Macroentities From Root Nodes** On the contrary, it may happen that the only root entity of an IS-A relationship represents an interesting concept for the hypertext: the leaf nodes are distinguished by subtleties and do not characterize any autonomous concept (in the hypertext perspective).

In this case, the root entity generates a macroentity, which possibly includes attributes that express the properties outlined by the original ER hierarchy.

Again from the hospital example, consider the ER hierarchy shown in Figure 4.2: it classifies physicians, on the basis of their specialization. Now, if we prefer that our hypertext represents physicians in general, we generate the only macroentity **PHYSICIAN**, adding it an attribute that specifies specialization, as shown in Figure 4.2.

Clearly, attributes and directed relationships coming from the leaf nodes must be treated as optional (their minimum cardinality is 0). For example, a (possible) relationship that connects physicians to surgical operations occurs only for those instances of macroentity **PHYSICIAN** that model surgeons.

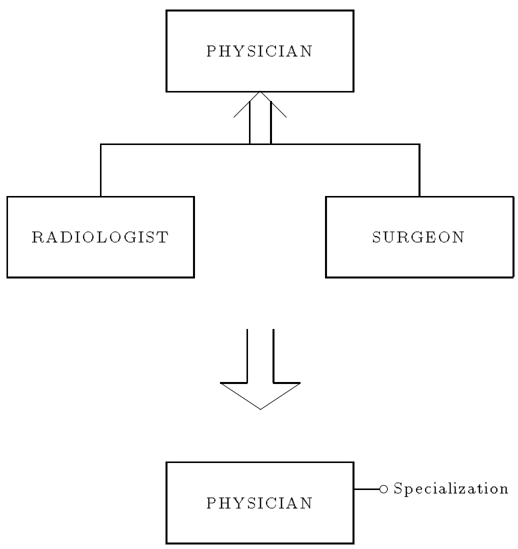



Figure 4.2: Deriving Macroentities from the root node of IS-A hierarchies

**Deriving Roles from Hierarchy Networks** As pointed out by several authors [12, 13, 37, 34], in conceptual data scheme hierarchy networks naturally arise in modeling complex realities. Consider for example Figure 4.3: it shows two interconnected IS-A relationships that (in a simplified way) model people involved in activities of a university clinic. In particular, physicians are both part of the caring personnel, and researchers: this twice participation suggests the twofold role of each physician. If designer decides to map each leaf node into a macroentity (and in this case the conditions discussed above may occur), then, for the **PHYSICIAN** macroentity, a roles partition can emphasize the researcher and the caring roles of physicians. Figure 4.4 illustrates the

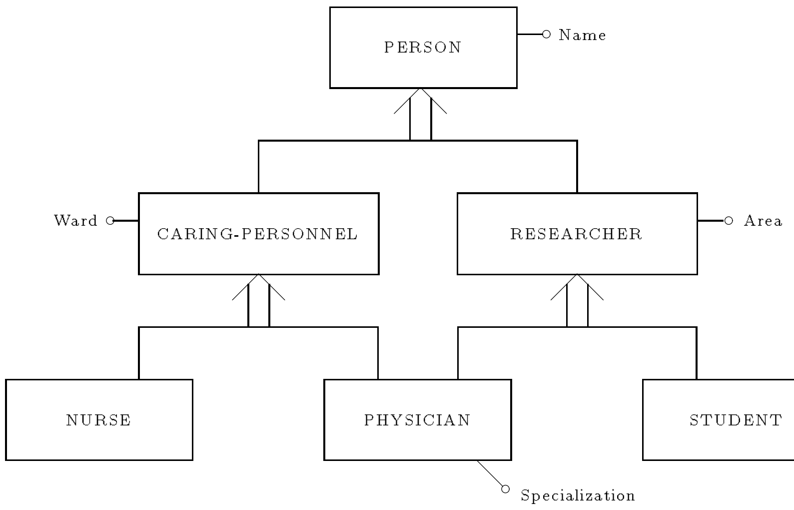



Figure 4.3: Entity **PHYSICIAN** participates two IS-A hierarchies

mapping of the ER scheme of Figure 4.3: each leaf entity generates a macroentity; in particular entity **PHYSICIAN** origins a macroentity with two roles, due to its participation with two distinct hierarchies. The first partition describes the caring role of physicians, and is named **CARE-PHYSICIAN**, the other refers physicians as researcher, and is named **RES-PHYSICIAN**.

Note that, beside attributes coming from the higher entity (**PERSON**), which migrate in all the generated macroentities, with respect to macroentity **PHYSICIAN**, we have that attributes from the originating entity become common to both role partitions, since they describe general properties of physicians. On the other side, attributes describing the specific roles are mapped from entities **RESEARCHER** and **CARING-PERSONNEL**.

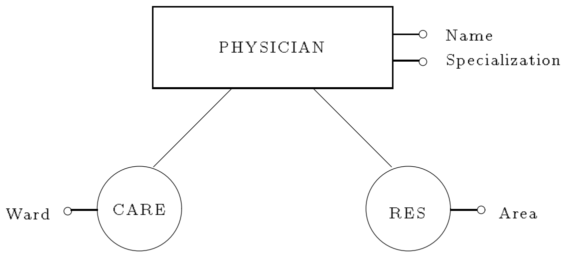



Figure 4.4: Mapping entity **PHYSICIAN** of Figure 4.3 in a macroentity with two roles

**Describing Macroentities by Merging Entities**

Describing macroentities by merging entities allows the attributes set of a macroentity to be enriched by properties that are spread over several entities and relationships of the ER starting scheme. In fact, this activity consists in collapsing relationships and entities into the originating entity, which identifies on the ER scheme the main domain concept of the corresponding macroentity.

Merging is recursively applied from the farthest entities and relationships to the originating entity, by mapping entities into composed (and possibly complex) attributes. Cardinality of relationships determines the cardinality of the generated attributes.

Figures 4.5 and 4.6 illustrate some cases of macroentity descriptions by entity merging. In Figure 4.5 macroentity `PHYSICIAN` is created from the homonymous ER entity; attributes are added by mapping entity `LOCALIZATION` into a composite attribute, which is then added to the generating entity. Figure 4.6 shows the generation of macroentity `COURSE`: first entity `ROOM` is merged into entity `LESSON`, which is in turn merged in `COURSE` as a complex attribute; attributes from entity `BOOK` complete the description of macroentity `COURSE`.

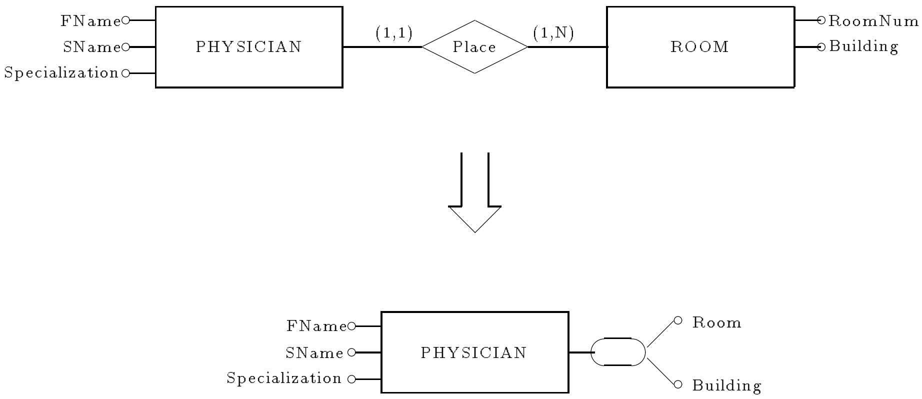



Figure 4.5: Describing macroentity `PHYSICIAN` by merging

### 4.1.2 Directed Relationships Design

Directed relationships are intensional descriptions of navigation paths based on conceptual associations between macroentities. The objective of directed relationships design phase is to identify, in the ER starting scheme, those concepts that represent associations between macroentities, and thus to properly map them into NCM directed relationships.

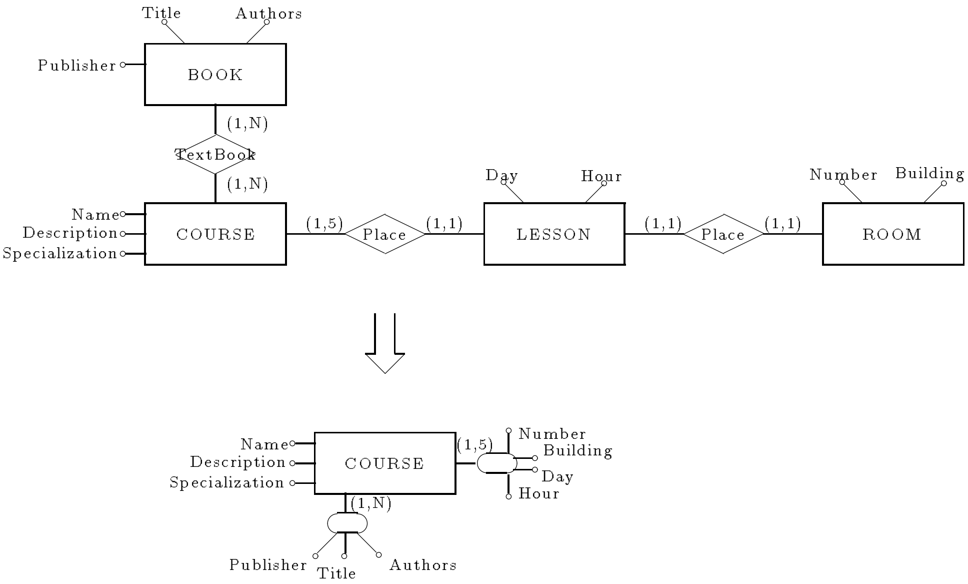



Figure 4.6: Describing macroentity `COURSE` by merging

An ER relationship involving a bunch of originating entities may naturally give rise to a directed relationship between the corresponding macroentities. However, it may happen that large portions of the starting ER scheme represent, in the hypertext perspective, associations between the main concepts, which are modeled by macroentities; thus, the design of a directed relationships might also proceed by mapping several relationships and entities into an NCM directed relationship. Moreover, since the design of directed relationships, at the instance level, deals with the design of navigation paths between the instances of macroentities, this activity also concerns the introduction of union nodes.

In the following, we use some examples in order to discuss the design of directed relationships, and the introduction of union nodes as well.

### Deriving Directed Relationships from Relationships

Each ER relationship that connects a set of originating entities usually produces an NCM directed relationship that associates the corresponding macroentities. Attributes that (possibly) describe the ER relationship are mapped as attributes of the directed relationship it has given origin.

The specification of the destination node is a task that usually deals with subjective decisions by designers: it describes how it is possible to navigate the path. Once the direction of traversal has been defined, cardinality constraints are fixed equaling cardinalities from the ER construct. [1]

For example consider Figure 4.7: the ER relationship `Responsible` between the generating entities `PHYSICIAN` and `PATIENT` is mapped into an NCM directed relationship.

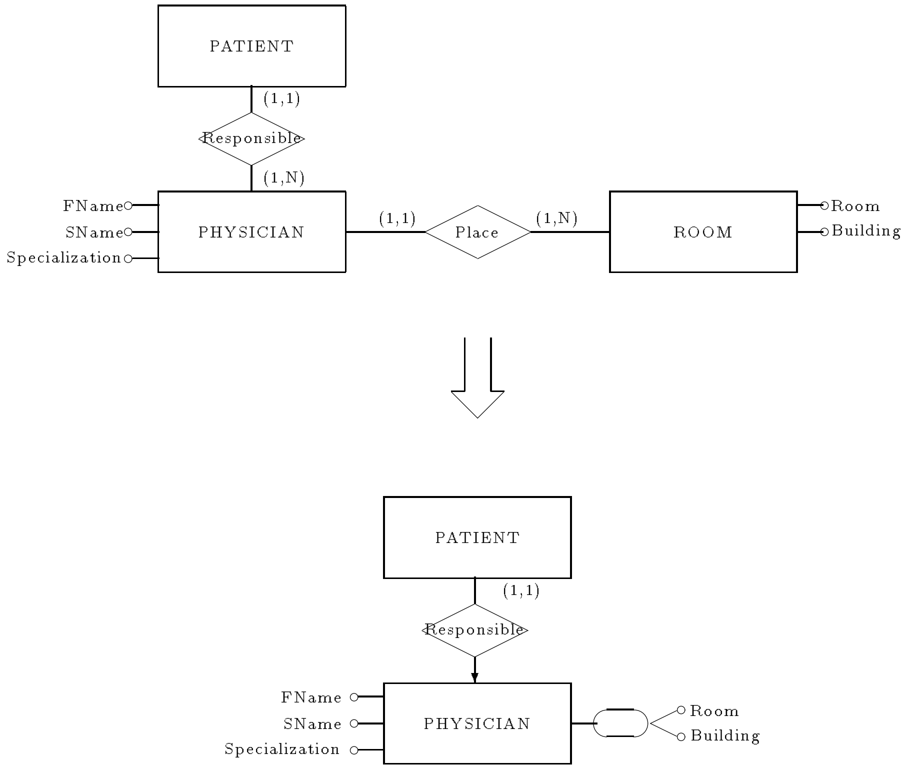



Figure 4.7: Deriving NCM directed relationships from ER relationships

### Deriving Directed Relationships from Entities

Usually, in the ER input scheme, there exist entities that do not generate any macroentity, but do represent associations between the concepts related to macroentities. Typically, such entities have been introduced in the ER scheme in order to achieve normalization.

For example, consider Figure 4.8: it shows an ER scheme that models how physicians co-operate in performing operations. Now suppose that, in translating this ER scheme into NCM emphasis goes on physicians and operations,

[1] Actually, whenever one of the originating entities come from an IS-A hierarchy, the lower cardinality could be forced to 0, as discussed above.

thus designer generates macroentities **PHYSICIAN** and **OPERATION**. In this perspective, entity **TEAM**, together with relationships **Member** and **Performs**, acts as an association that relates such macroentities, therefore it is mapped into the NCM directed relationship **Team**.

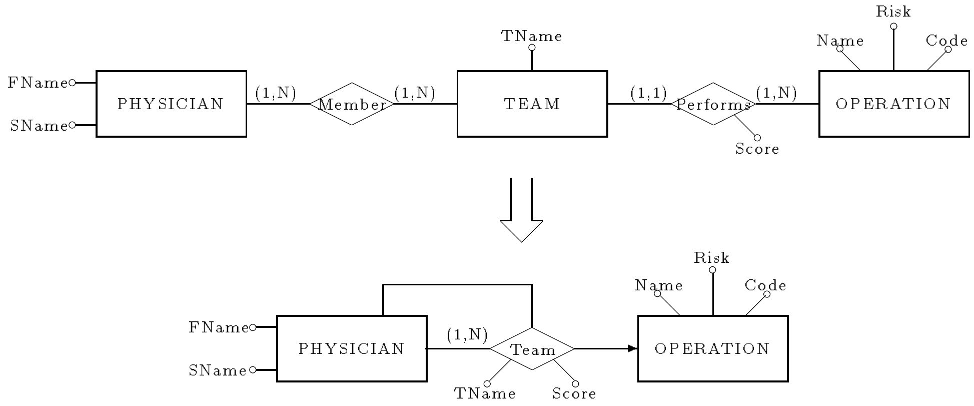



Figure 4.8: Deriving NCM directed relationships from ER entities

Note that cardinality constraints over relationship **Member** express that several physicians are member of a given team. Designer can map this property into the NCM scheme, transforming the binary directed relationship **Team** into a ternary directed relationship, where macroentity **PHYSICIAN** participates twice: the self relationship models that each physician collaborates with several other colleagues to perform an operation. Figure 4.8 shows also the generated NCM scheme.

### Introducing Union Nodes

During the design of directed relationships, designers may also decide to introduce union nodes, in order to improve navigation features.

As we discussed above, union nodes may arise from IS-A hierarchies. Specifically, if there is a relationship in the ER scheme involving as destination node an entity that is the root of an IS-A hierarchy whose children was mapped into macroentities, then it should be considered whether a union node could be useful, in order to allows for navigations from (another) macroentity to the various "child" macroentities.

Let us consider again example discussed in Figure 4.3. Assume that the entity **RESEARCHER** participates to the relationship **WorksIn** with the entity **PROJECT**, as shown in Figure 4.9. Also, suppose designer has to model a navigation path that starts from projects and leads to members. Now two alternatives exist in order to translate the relationship into NCM. On the one hand, two directed relationships might arise: this solution stresses for each member the class of destination yet in the source node; Figure 4.10 illustrates this case. On the other hand, in order to have a unique class of links that connect projects to their member, a union node might be preferred, as shown in Figure 4.11.

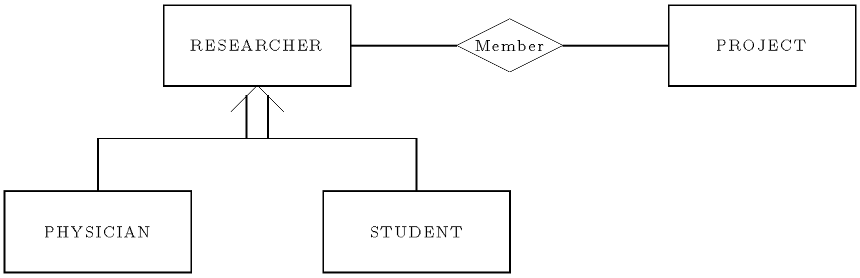



Figure 4.9: Designing directed relationships when dealing with IS-A hierarchies

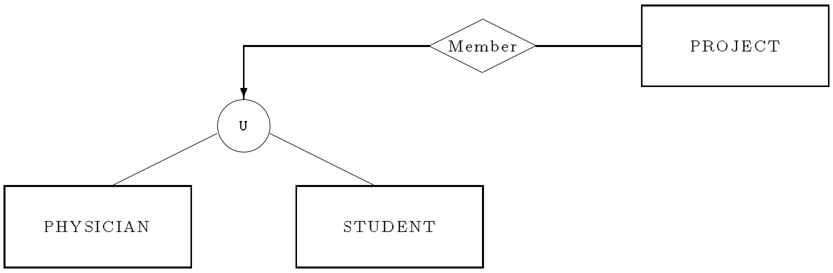



Figure 4.10: Introducing Union Nodes from the root nodes of IS-A hierarchies

### 4.1.3 Aggregation Design

In this phase, which aims at mapping the starting ER scheme into an NCM skeleton, aggregations are defined on the basis of IS-A hierarchies that generate macroentities from leave nodes.

Consider for example Figure 4.3. Entity `RESEARCHER`, which does not give rise to any macroentity, models a conceptual classification of researchers; therefore, it makes sense to define an NCM aggregation that gathers links to access instances of macroentities `PHYSICIAN` and `STUDENT`, as shown in Figure 4.12. Whenever the destination node of an aggregation is a macroentity that has been derived from the root entity of an IS-A hierarchy, it may be useful to partition aggregation links, in order to aggregate subsets of the destination

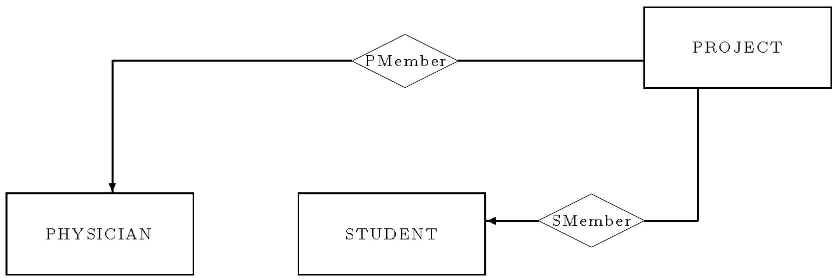



Figure 4.11: Introducing a twice directed relationships

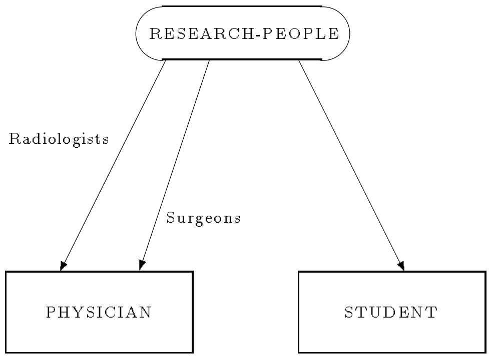



Figure 4.12: Deriving NCM aggregations

macroentity separately. As an example consider again Figure 4.12: two aggregation link go to physicians: the former reaches "surgeons", the latter "radiologists". Note that this operation is based on conceptual properties expressed in the starting ER scheme (see Figure 4.2.

## 4.2 ER-NCM Translation Primitives

The process of mapping ER schemes into NCM could be rather complex and involve several actions. This section presents a set of tools we propose for documenting each of the actions that leads to generate an NCM construct. We introduce a set of primitives, we call ER-NCM *translation primitives*, that allow for the description of the activities that give rise to the NCM scheme starting from the starting ER scheme.

We observe that the whole mapping process can be described by means of two sets of primitives that *(i)* operate transformations over the ER scheme, and *(ii)* generate NCM constructs. Primitives belonging to the former set have an ER scheme as input, and produce an ER scheme; they perform transformations such as the flatting of IS-A hierarchies into entities, or the merging of entities. The latter set includes primitives that aims at generating NCM constructs, such as the originating of a macroentity, or the creation of an aggregation; an ER scheme represents the input of these primitives, and an NCM construct is their output.

We group both these primitives into a unique set, the set of ER-NCM translation primitives, since the generation of each NCM construct usually requires their heterogeneous application.

In the following we briefly present the set of ER-NCM translation primitives that are most commonly used.

1. Primitive $T_1$ generates a macroentity from an ER entity. All the attributes describing the resulting macroentity come from a subset of the attributes of the originating entity.

2. Primitive $T_2$ flattens as IS-A hierarchy into an entity; attributes are possibly introduced in order to suitably describe properties of the leaf entities.

3. Primitive $T_3$ generates a macroentity with some roles from an ER complex hierarchy.

4. Primitive $T_4$ transforms an entity and a relationship into a composite attribute. Note that, the cardinality of the generated attribute equals cardinality of the relationship.

5. Primitive $T_5$ creates a directed relationship from an ER relationship.

6. Primitive $T_6$ transforms an entity and two relationships into an N-ary directed relationship. The arity of the generated relationship may depend on cardinalities of the involved relationships.

7. Primitive $T_7$ creates a union node from the root entity of an ER hierarchy.

8. Primitive $T_8$ creates an aggregation from the root entity of an ER hierarchy.

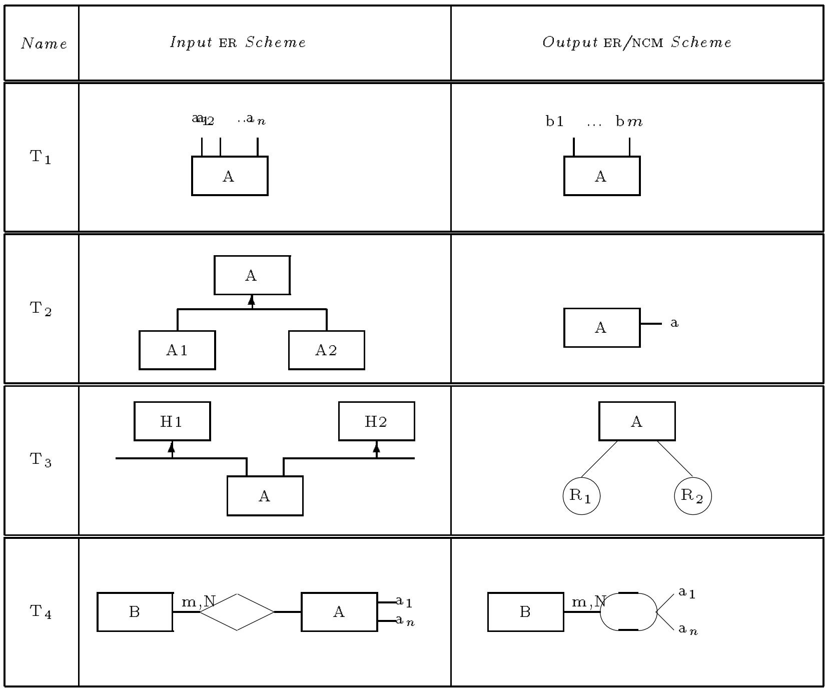



Figure 4.13: ER-NCM Translation Primitives $T_1$, $T_2$, $T_3$, and $T_3$ are used to generate macroentities

Figure 4.13 illustrates the ER-NCM Translation Primitives that allow for the generation of macroentities from ER entities. Primitives $T_5$, $T_6$ and $T_7$ support the design of navigation paths based on conceptual relationships, and are illustrated Figure 4.14. Primitive $T_8$ aims at building the hypertext access structure by introducing aggregations, and is depicted in Figure 4.15. On the basis of the ER-NCM translation primitives, phases we have discussed in 4.1.1, 4.1.2 and 4.1.3 are effectively described by the following scheme:

PHASE 1 *Generate macroentities:* identify originating entities, i.e. entities that express the main domain concepts to be presented in the hypertext; for each originating entity:

- if it does not participate any IS-A hierarchy, or it is a leaf node of an IS-A hierarchy, then apply primitive $T_1$; else,
- if it is the root node of an IS-A hierarchy, then apply primitive $T_2$, or, possibly, create a role partition by applying $T_3$.

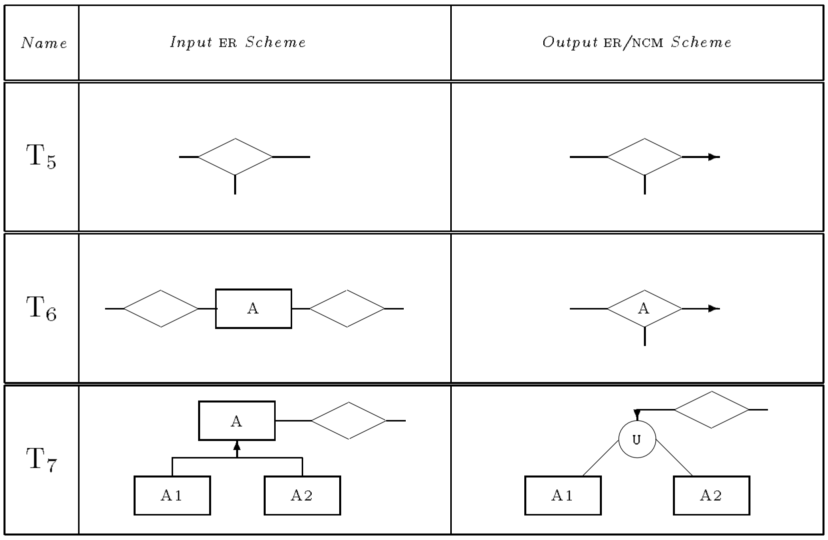



Figure 4.14: ER-NCM Translation Primitives $T_5$, $T_6$, and $T_7$ aim at introducing directed relationships

- *Describe Macroentities:* (possibly) apply $T_4$ in order to add attributes from farther entities.

PHASE 2 *Introduce Directed Relationships:*

- for each entity that describes a conceptual association between originating entities apply $T_6$;
- for each ER N-ary relationship involving N originating entities, apply $T_5$;
- for each ER relationship involving an entity that participates an IS-A relationship apply $T_5$, and possibly introduce a union node by $T_7$.

PHASE 3 *Introduce Aggregations:* apply $T_8$ for each entity that *(i)* is the root of an IS-A relationship, *(ii)* is not a generating entity.

## 4.3 Refining a Navigation Conceptual Scheme

The mapping process concludes with aggregation design and results in an NCM skeleton scheme. The organization of information reflects properties of the application domain, and the structure of navigation paths arises from associations and relationships based on conceptual properties. However, the

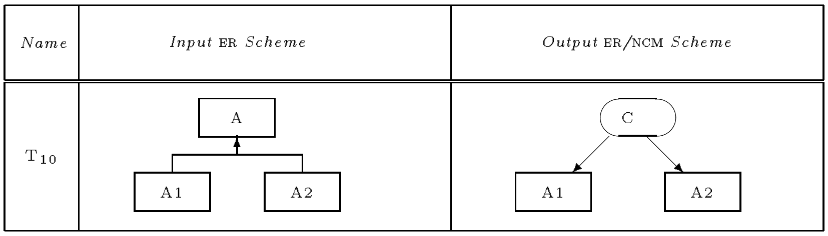



Figure 4.15: ER-NCM Translation Primitive $T_8$ allows for the introduction of aggregations from IS-A hierarchies

skeleton scheme may present some limitations. In particular information may not be sufficiently organized, and further navigation paths may be required in order to access information: new directed relationships may be introduced in order to provide significant paths that did not arise from the ER scheme (since redundant relationships usually do not occur in conceptual scheme); also, the structure of aggregations may need several restructuring, in order to better access paths; finally, roles can better organize pieces of information in some macroentity, and union nodes provide more effective navigation paths.

In the following we introduce a set of transformation primitives, we call NCM Refinement Primitives, which are tools at disposal of designer to restructure NCM schemes. Refinement primitives apply to an input NCM scheme $S_1$ and produce a resulting NCM scheme $S_2$. Figure 4.16 illustrate the set of NCM refinement primitives.

- Primitive $R_1$ creates some roles in a macroentity. Using this primitive corresponds to recognize that a macroentity describes a class of objects that plays different roles in the hypertext, and that different navigation paths should be associated for each role.

- Primitive $R_2$ partitions an aggregation link (whose destination node is a macroentity) based on some properties of the destination macroentity, and thus aims at improving the hypertext access structure.

- Primitive $R_3$ introduces an aggregation. Note that, $R_3$ can aggregate both macroentities and aggregations. This primitives is applied in order to enhance the of the access structure.

- Primitive $R_4$ introduces a new directed relationship to connect two macroentities; the application of this primitive allows designer to better the navigation between concepts.

- Primitive $R_5$ adds a macroentity to an aggregation. This primitive aims at giving a better access to the instances of the involved macroentity.

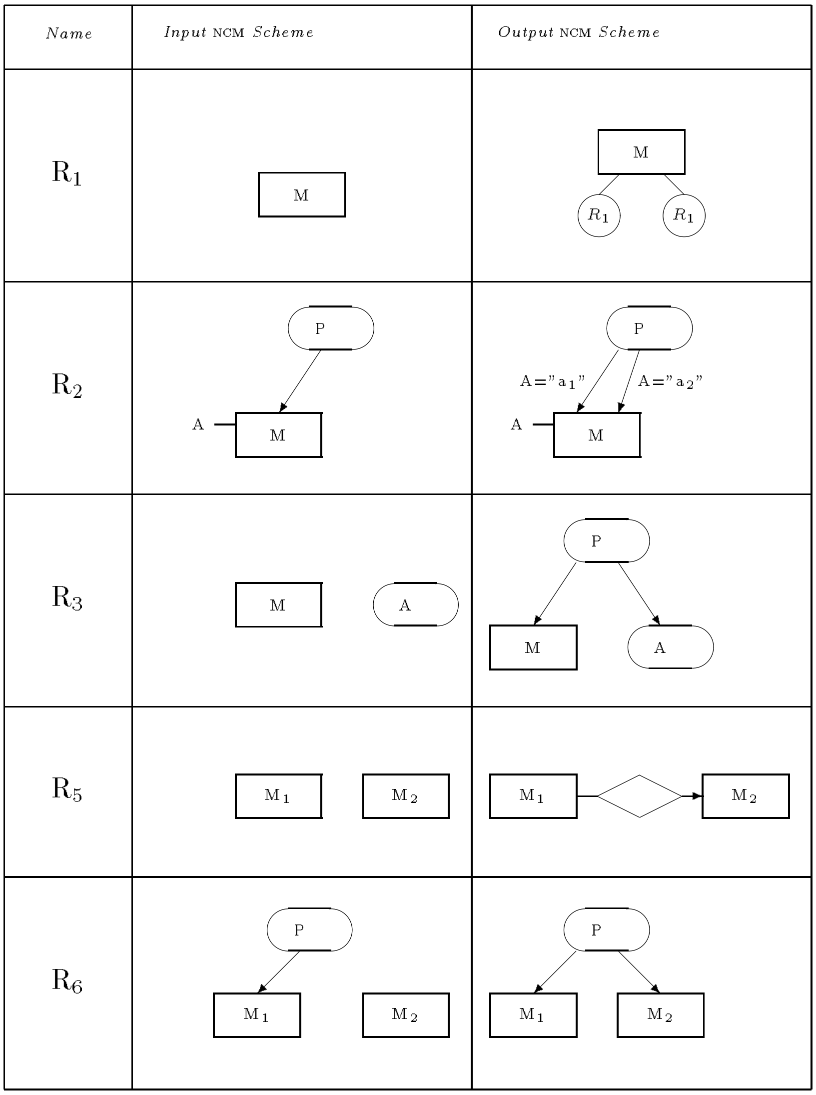



Figure 4.16: NCM Refinement Primitives

# Chapter 5

# Hypertext Logical Design

The main objective of *Hypertext Logical Design* is to produce an ADM scheme, starting from the requirements expressed in the conceptual scheme of navigation. The approach we present is similar to that of the conceptual design process: first we map the NCM scheme into an ADM skeleton; then a restructuring process arises in order to better the logical organization of data. The mapping process consists in a thorough and systematic translation: NCM nodes (macroentities, roles and aggregations) are mapped into page-schemes, and NCM links (directed relationships, role links and aggregation links) into ADM link attributes. The restructuring process holds the requirements of the conceptual description, and aims at refining the structure of data in order to improve both the effectiveness of navigation paths and the organization of information.

Since the need for restructuring the logical organization of data in the hypertext may occur during its whole life-cycle, our restructuring process has to be considered also as a tool for maintenance.

The chapter is organized as follows: Section 5.1 illustrates the process that maps NCM specifications into an ADM scheme. Section 5.2 discusses how the hypertext can be improved, reasoning on the ADM scheme.

## 5.1 Mapping NCM schemes into ADM

A logical specification of the hypertext bridges the gap between the high level description supported by a conceptual model of navigation, and the actual organization of data in HTML pages. On the one hand, an ADM scheme abstracts the logical organization of data inside HTML pages describing their contents in terms of attributes, and depicts the overall organization of hypertext as a graph of pages connected by links. On the other hand, an NCM scheme provides an hypertextual description of the application domain: macroentities capture the main concepts of the universe of discourse, directed relationships express the navigation paths between them on the basis of conceptual associations, and aggregations describe the access structure, getting inspiration from conceptual aggregations and classifications.

The design of an ADM scheme aims at organizing information in pages on the basis of the conceptual properties expressed in the conceptual scheme of the navigation. Thus first phase of the logical design of the hypertext creates an ADM scheme starting from an NCM description: the process consists in mapping each NCM construct into ADM, on the basis of three main steps:

STEP 1 Map Macroentities into page schemes

STEP 2 Map directed relationships into link attributes

STEP 3 Map aggregations into page-schemes

In the following we discuss in detail each step.

### 5.1.1 Mapping Macroentities

There are two main ways of organizing the presentation of a macroentity instance using Web pages. The most natural way is to use one page for each macroentity instance. However, there exist cases in which this solution might not be appropriate, and the designer can decide to represent all instances of the macroentity in a single page, as items of a list. Usually, this solution is applied when instances have a particularly concise description; nevertheless, there is no definitive rule to choose one of the previous strategies, since some macroentities, although with concise instance description, might represent concepts that the designer might prefer to represent using one page for each instance.

Thus, when mapping macroentities to page-schemes, a first design choice consists in deciding which macroentities will be translated following the "one-page-per-instance" strategy, and for which one a single page listing all instances is more appropriate. In the former case, a page-scheme with the same attributes as the macroentity is generated. In the latter case, the resulting page-scheme consists of a list of tuples having the same attributes as the originating macroentity. When translating attributes, NCM single valued attributes are mapped to ADM simple attributes (text or images), and NCM multi-valued attributes are mapped to ADM lists. Also, attributes whose minimum cardinality is 0 are labelled as optional.

Let us illustrate two examples: Figure 5.1 shows how the macroentity `RESEARCH-GROUP` is mapped into ADM, in the case we decide to present each instance in separate pages, one for each research group; note that, the multi-valued attribute `Topic` has been mapped into a list attribute. Figure 5.2 depicts macroentity `SEMINAR` and its mapping into a list of tuples.

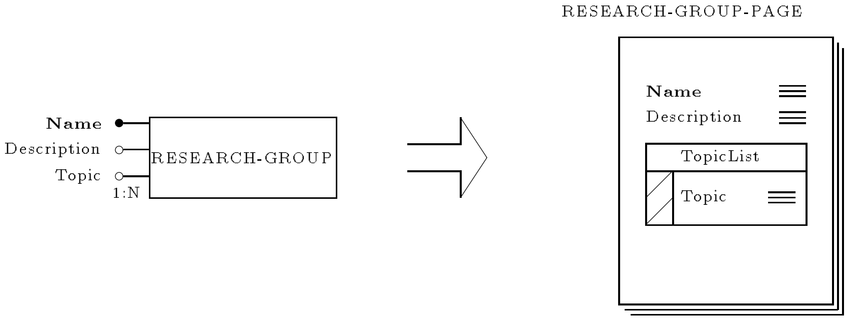



Figure 5.1: Mapping macroentity `RESEARCH-GROUP` into ADM page-scheme `RESEARCH-GROUP-PAGE`

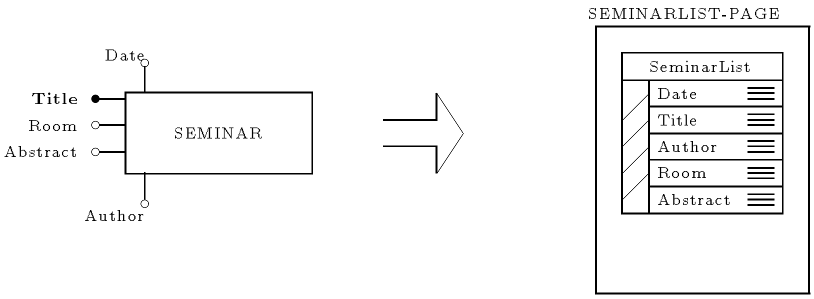



Figure 5.2: Mapping macroentity `SEMINAR` into an ADM list

### Mapping Roles

Macroentities with roles are naturally translated in ADM, by mapping each role node into a non-unique page-schemes. [1] The attributes set of a page-scheme corresponding a role is made by both the attributes from the NCM role node, and the attributes that are shared with other roles of the same macroentity. Role links are mapped into ADM link attributes: a link is added into the page-scheme that corresponds to the source role; anchor of the link is a constant, which expresses the role described by the the target page-scheme, and reference is the target page-scheme; for each shared attribute, a link constraint is associated to the link , and expresses that, whenever the link is

[1] Since roles are used to partition macroentities describing complex concepts, it is not appropriate to map them into lists.

crossed, attributes in the source page-scheme equal attributes in the target page-scheme.

Figure 5.3 shows the mapping of macroentity **PHYSICIAN**: note that the roles **RESEARCHER** and **TEACHER** are mapped into page-scheme **PHYSICIAN-RES-PAGE** and into page-scheme **PHYSICIAN-EDU-PAGE**, respectively.

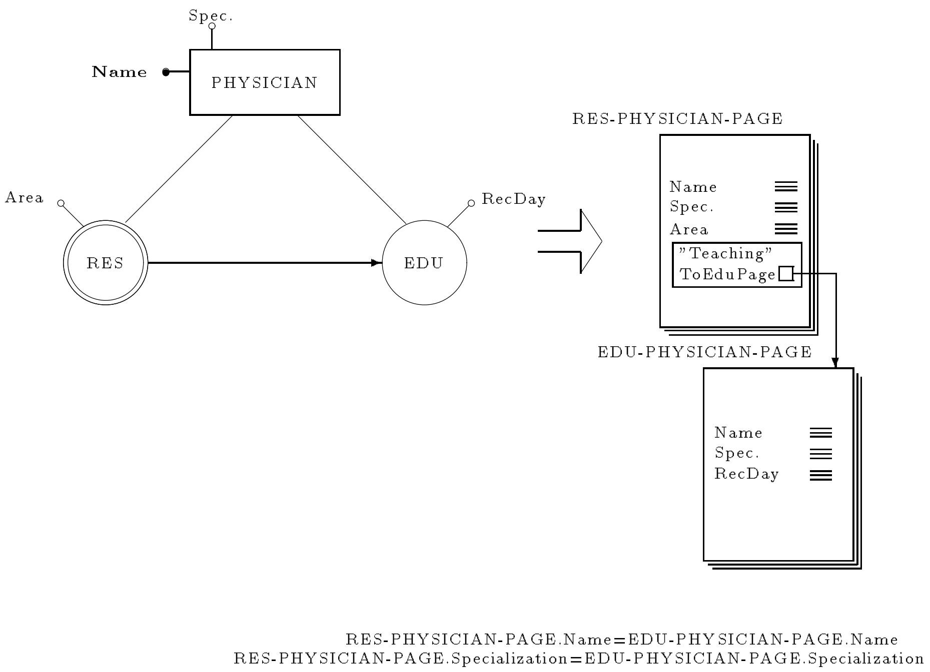



Figure 5.3: Mapping roles of macroentity **PHYSICIAN** into ADM

### 5.1.2 Mapping Directed Relationships

As for role-links, directed relationships are mapped into ADM links between page schemes. However, this step depends on several factors, such as the cardinality of the relationship, its arity, and its (possible) attributes; moreover, also the way in which macroentities participating in the relationships have been mapped into ADM influences this step.

The mapping into ADM of a directed relationship $DR_i$ proceeds by considering each macroentity $M_j$ that participates in $DR_i$ as source node, and adding to the corresponding ADM page-scheme (list) $P_J$ ($L_j$) a set of attributes, as follows:

- if the maximum cardinality of macroentity $M_J$ in the directed relationship $R_i$ is at most one, then:

1. as many link attributes as the number of the destination macroentities are added to $P_j$ $(L_j)$ :
   - anchor of each link is the attribute corresponding to the descriptive key of the destination macroentity,
   - reference is the page scheme that has been created from the destination macroentity;
   - if the destination macroentity has been mapped into ADM as a list of tuple, then an offset must allow the link to point (inside the destination page scheme) the tuple of the list;
   - a link constraint documents that the anchor attributes must equal the attributes of the destination page-scheme that correspond to the imported descriptive keys;
2. all the attributes of the directed relationship are mapped into ADM and added to $P_j$ $(L_j)$;
3. as many ADM attributes as the number of the other macroentities $M_k$, with $k = 1 \ldots N, K \neq j$, participating in the directed relationship $R_i$ as source nodes are added to $P_j$ $(L_j)$; each attribute is the mapping into ADM of the descriptive key of macroentity $M_k$.

- if the maximum cardinality of $M_j$ in $R_i$ is greater than one, then a list of tuples having the previous attributes is added to $P_j$ $(L_j)$.

Let us discuss now some examples concerning the mapping of directed relationships into ADM. In Figure 5.1.2 the directed relationship `Responsible`, whose upper cardinality is at most one, is mapped into a link attribute in the source page-scheme `SEMINARLIST-PAGE`, which corresponds to the macroentity `SEMINAR`. Note that a link attribute (`ToPhysician`) is introduced into the list attribute `SeminarList`.

Figure 5.1.2 illustrates the mapping of a directed relationship into a list of links: since the macroentity `RESEARCH-GROUP` is in the directed relationship `Member` with maximum cardinality $N$, thus a list of link occurs in the ADM page scheme `RESEARCH-GROUP-PAGE`.

Finally, in Figure 5.1.2, we show how the ternary directed relationship `Diagnosis` is treated: a list of item is added to each ADM construct that correspond to the source macroentities; each item is a tuple with *(i)* a link attribute to the destination page-scheme `CLINIC-PAGE`, and *(ii)* the descriptive key of the other source macroentity.

### Mapping Union Nodes

When the destination of a directed relationship is a union node, first the union node is mapped into an ADM heterogeneous union, then the same procedure

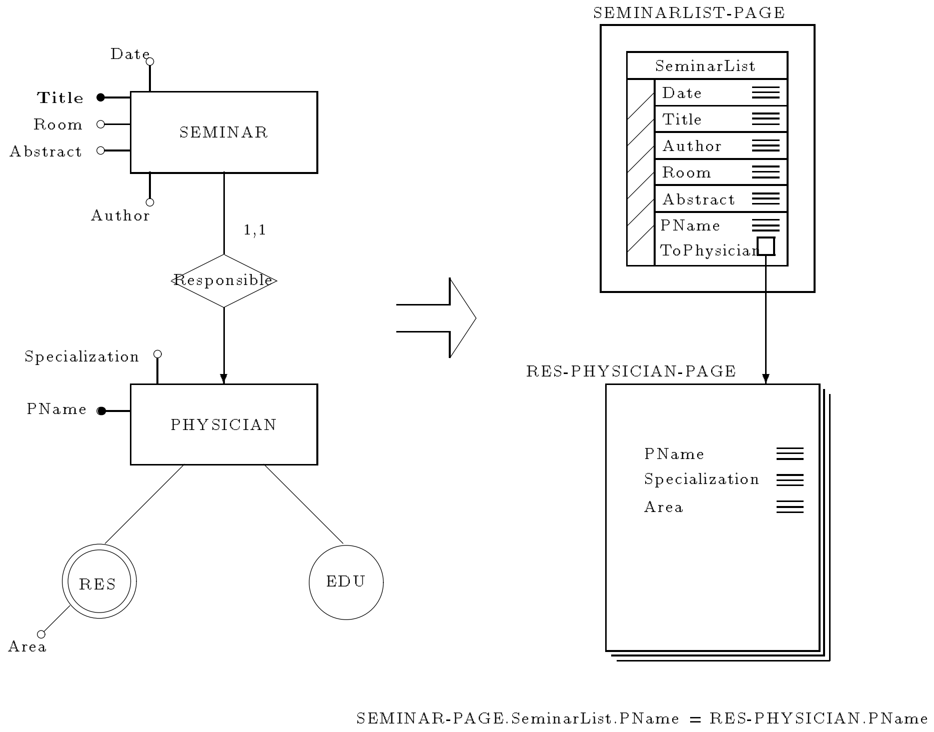



Figure 5.4: The NCM directed relationship `Responsible` is mapped into an ADM link attribute in page-scheme `SEMINARLIST-PAGE`

described above can be applied. However, some problem may arise if the descriptive keys of macroentities involved in the union node differ one each other.

Now, we argued that the descriptive key of the target construct has to be imported as the anchor of a link attribute in the source page-scheme. Then, if the descriptive keys of macroentities involved in a union differ, we need to model the possibility that different attributes arise as anchor. In order to keep the flexibility expressed into the input NCM scheme, we can add all the attributes corresponding to the descriptive keys of the destination macroentities; however, because of the semantics of the union node, which leads a link to the disjoint union of its macroentities, they are mutually exclusive, and thus we have to label them as optional.

### 5.1.3 Mapping Aggregations

Each aggregation node is mapped into an ADM unique page-scheme. NCM aggregation links correspond to link attributes with some subtleties, which depend on the destination node, as follows:

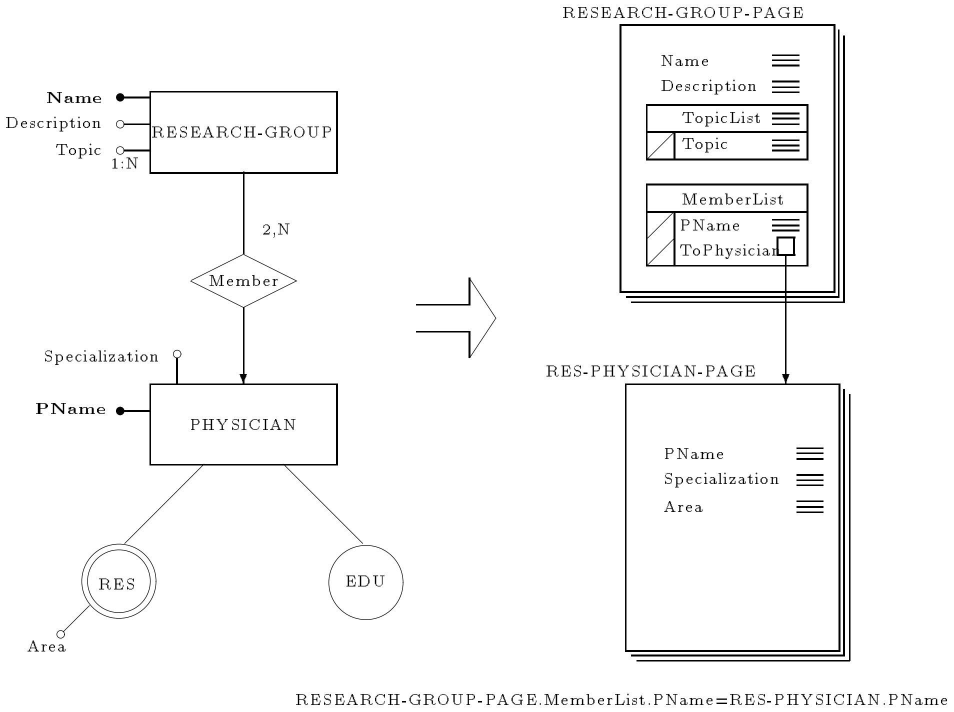



Figure 5.5: The NCM directed relationship `Member` is mapped into an ADM list in page-scheme `RESEARCH-GROUP-PAGE`

- if the destination node is in turn an aggregation node then a single link attribute to the corresponding page-scheme is used: the anchor is a constant string, describing the concepts which underlies the destination page-scheme;
- if the destination node is macroentity mapped into a list then a link attribute to the corresponding page-scheme is used. Nevertheless, if the aggregation link defines a partition over the destination macroentity, thus, the destination page-scheme is not unique anymore (since it has to be partitioned), and a link constraint defines the partitioning predicate;
- if the destination node is a macroentity, mapped into a page-scheme or a union node, an ADM list attribute is added to the source page-scheme: each item of the list is a link to one instance of the destination page-scheme (anchor, reference and link constraint arise in the usual way).

Figure 5.7 and 5.8 illustrate this step: aggregation `RESEARCH` is mapped into the unique page-scheme `RESEARCH-PAGE`; the component node `PEOPLE` is in turn an aggregation node: it is mapped into a unique page-scheme, `PEOPLE-PAGE`,

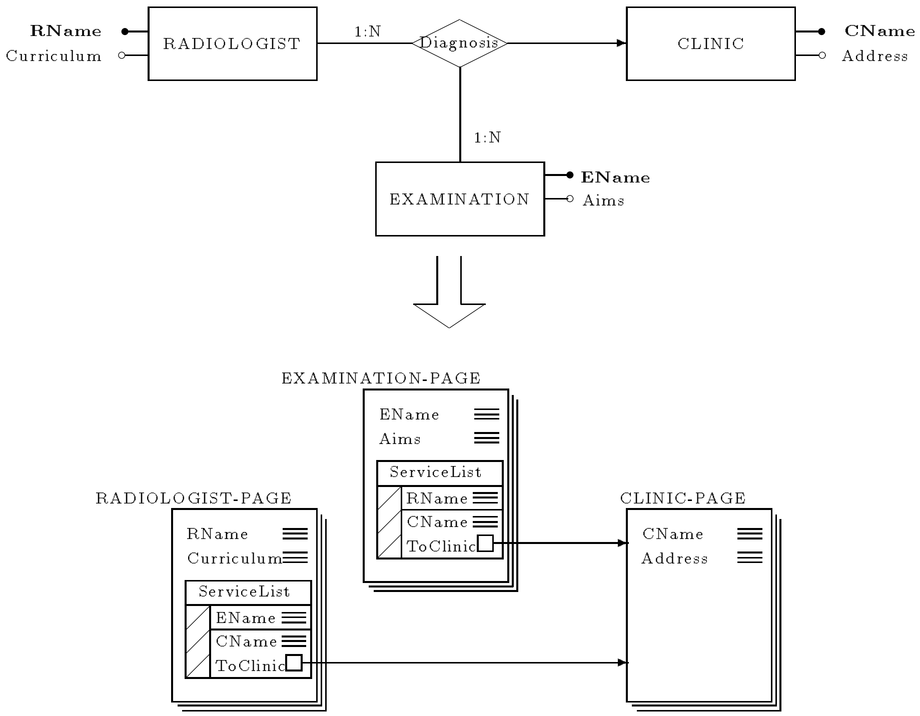



Figure 5.6: The NCM ternary directed relationship `DiagnosticService` is mapped into ADM

and a link attribute is added to page-scheme `RESEARCH-PAGE`; the component node `SEMINAR` corresponds to a macroentity that is mapped into an ADM list in the unique page-scheme `PUBLICATION-LIST-PAGE`, thus a link attribute is added to the page-scheme `RESEARCH-PAGE`; finally, let us consider the component node `RESEARCH-GROUP`: it is a macroentity, and in ADM it corresponds to the page-scheme `RESEARCH-GROUP-PAGE`: since it is a non unique page scheme, a list of links has been added to the `RESEARCH-PAGE` page-scheme in order to model the aggregation link.

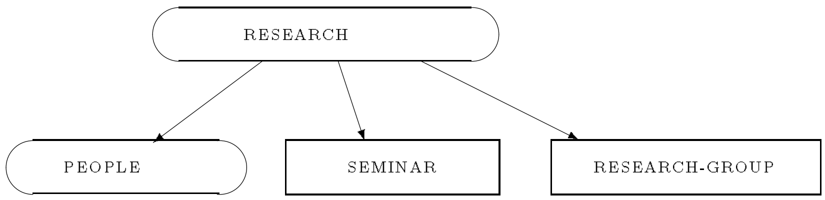



Figure 5.7: The aggregation `RESEARCH`

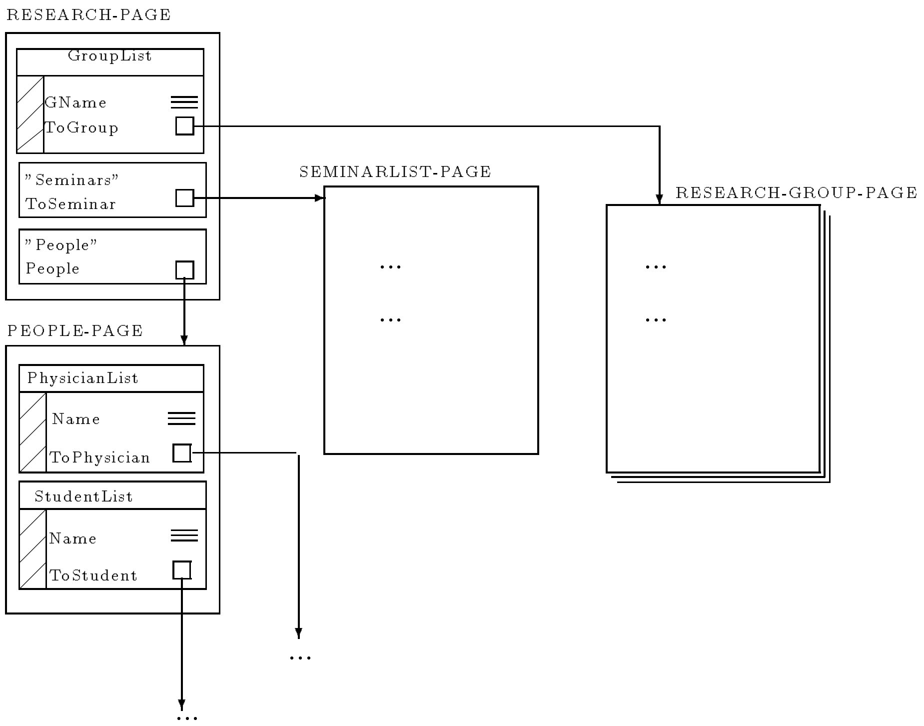



Figure 5.8: Mapping NCM aggregations into ADM

## 5.2 Restructuring ADM schemes

After all NCM primitives have been mapped into ADM constructs, based on the steps discussed in Section 5.1, a first ADM skeleton results. Although this logical scheme somehow reflects the conceptual NCM scheme, it may present limitations that originate several problems. The following is a list of the classes of problems that more frequently occur:

1. some of the page schemes can be considered too dense, and thus hardly effective to browse;
2. complex or counterintuitive navigation paths may be required in order to access information; for example, lists of links may result too long and not enough structured;
3. data may not be sufficiently organized; for example lists whose items correspond to macroentity instances usually are very large, and they can be ordered following only a given criterion.

Usually, the skeleton scheme that is obtained by the mapping process presents these problems; however, they may arise during the hypertext life-cycle as well, due to updates on the extension of data. Therefore, since it is often the case that the need for changing the organization of a hypertext arises, still with respect to the same conceptual structure, the issue of restructuring a scheme is relevant both in a standard design activity and in a maintenance initiative.

ADM provides a natural framework to reason about page organization, and to re-organize data without changing the hypertext conceptual structure. In the following, we discuss some classes of transformations, called ADM *transformation primitives*, that can be used to restructure an ADM scheme; each transformation takes an ADM scheme as input and produces an ADM scheme as output; moreover, for each of these transformations, also the converse is possible.

### 5.2.1 Slicing page schemes

Page schemes corresponding to particularly rich macroentities may be split in two (or more) page schemes, each presenting a subset of the attributes of the page scheme; in this way, different perspectives of the same concept can be offered. This transformation takes a page-scheme as input and creates new page-schemes by performing a vertical (possibly overlapping) partition among its attributes. This is somehow reminiscent of the role creation during the conceptual design, and actually also the slicing of a page-scheme should be done following some conceptual criterion; nevertheless, here the motivations to perform this activity are due to logical properties of page-schemes, such as the presence of large lists of items, originated, for example, by the mapping of either multi-valued NCM attributes, or directed relationship. Clearly, whenever a designer decides to apply this transformation, entering links have to be properly set up, choosing which of the new page-schemes they have to point to. Also, the designer can define new links that connect the various partitions one each other.

### 5.2.2 Managing Lists of Links

A problem that frequently arises is the one of lists of links containing a very large number of items, that make very difficult to navigate from one page to the other. Usually, large list of links occur in page-schemes originated by mapping NCM aggregations, since, at the extension level an aggregation link has the same number of instances as the macroentity it leads to. However, large list of links may also arise in page-schemes that have been created from macroentities, due to the mapping of directed relationships.

In order to improve pages in which a large list of links occur, two alternatives are possible: *(i)* the list can be re-organized introducing different levels;

or *(ii)* it may be substituted by a form.

### Introducing Multi-Level Lists

Introducing multi-level lists consists in grouping items in the list on the basis of some criterion, and moving each group to a different (new) page; the original list will thus contain a fewer number of entries, i.e., one for each group of items.

Consider for example the page-scheme `PHYSICIAN-LIST-PAGE` in Figure 5.9: since the number of physician might be quite large, we can introduce a multi-level list, dividing physicians on the basis of their initial. In this way, the original page contains only one item for each initial, pointing to a page containing the list of all physicians with that initial. Figures 5.9 illustrates how this transformation has been applied on the physician list: note that the resulting scheme introduces a new page-scheme (`PHYSICIAN-INITIALS-PAGE`), and that the restructured page-scheme `PHYSICIAN-LIST-PAGE` is not unique anymore. Note that, as generalization of this transformation, one can consider

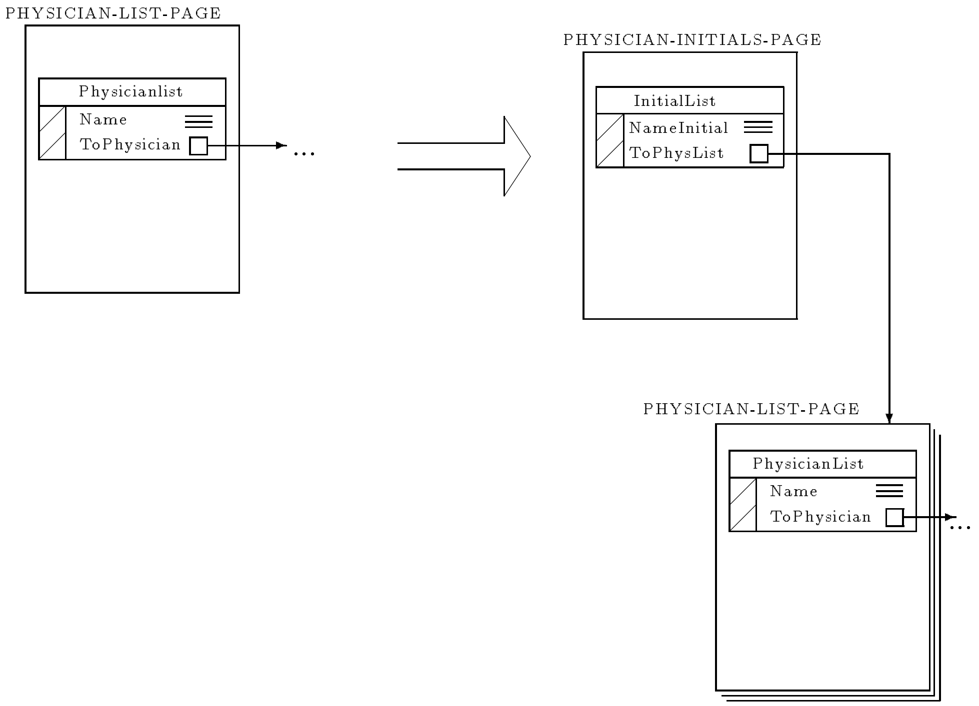



Figure 5.9: ADM transformation: Introducing Multilevel Lists

the whole list as a group, and move it into a new page-scheme, which is thus unique, leaving only a link attribute (that points to the new page-scheme) in substitution of the list.

### Introducing Forms

When dealing with a very long list of links, as an alternative to the introduction of multi-level lists, designers can use a form in substitution of the list; in this way, list items are not to be physically stored in the page, but they may be encoded using some program running on the server: when a request is made based on the form, the program is executed and only relevant items are returned. Figure 5.10 illustrates how the introduction of a form eliminates the list of links: `PatientList` may be huge if, for example, it have to point to all the patient that have been cared by the organization in the last ten years. Thus, a form is preferred to the list of links.

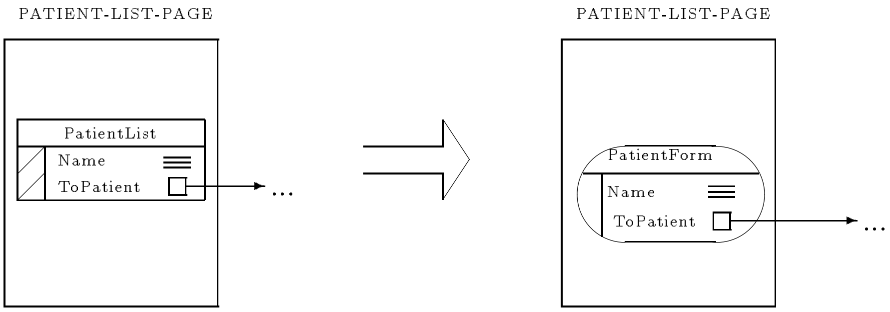



Figure 5.10: ADM transformation: Introducing Forms

## 5.2.3 Lists Horizontal Partitioning

We have seen that it may be appropriate to represent the instances of some macroentity, organizing them in a list, inside a unique page-scheme. However, it may be often the case that such a structure is not effective.

Let us illustrate some of the problems that arise by means of an example. Consider Figure 5.11: it shows the page-schemes `PHYSICIAN-PAGE` and `PUBLICATION-PAGE`; note that, each publication is presented as the item of a list, `PublicationList`, which is contained in a unique page; differently, one page is devoted to each physician. The attribute `PaperList`, in page-scheme `PHYSICIAN-PAGE` is an ADM list that contains links to the items of `PublicationList`. This way of organizing information and navigation paths may have several severe limitations. First, the instance of `PublicationList` might contain a very large number of items, since it provides information about all the papers written by all the physicians. Moreover, navigation paths between physicians and their publications is not effective for the user, due to the amount of links that have to be followed back and forth if one wants to browse the publications of a given physician (assuming, for example, that `PublicationList` is ordered by the name of the first author, and that several physicians can author a paper, it is not possible to group papers by author). Finally, observe that also `PaperList` might be rather large.

Some of the transformations discussed above could be applied, but the resulting scheme would continue to be unsatisfactory. For example, the list `PaperList` can be moved into a separate page-scheme, in order to lighten `PHYSICIAN-PAGE`; nevertheless, the browsing of `PublicationList` would continue to be very difficult.

*Lists horizontal partitioning* helps to improve the organization of information in lists. This transformation gets a unique page-scheme that contains a list of items, and results in a non-unique page-scheme that has a list attribute with the same structure as the original one: the instance of the list in each page corresponds to a selection (a horizontal partition) of tuples from the originating list. The selection predicate is associated with each link that leads to the new page-scheme, and is defined by means of link-constraints.

Figure 5.12 illustrates the horizontal partitioning of list `PublicationList`. Note that, *(i)* the resulting page-scheme `PUBLICATION-PAGE`, is not unique anymore; *(ii)* the link `ToPaper` in page-scheme `PHYSICIAN-PAGE` is associated with a link constraint defining that, following the link from the page of a given physician, one reaches the list of papers where, among the authors, the name of such a physician always appears.

Clearly, links that have not any link constraint refers to an instance containing the whole list.

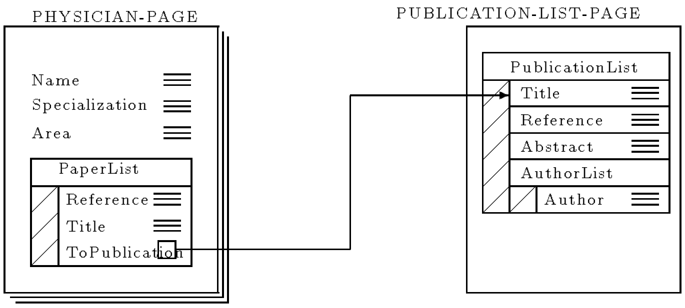
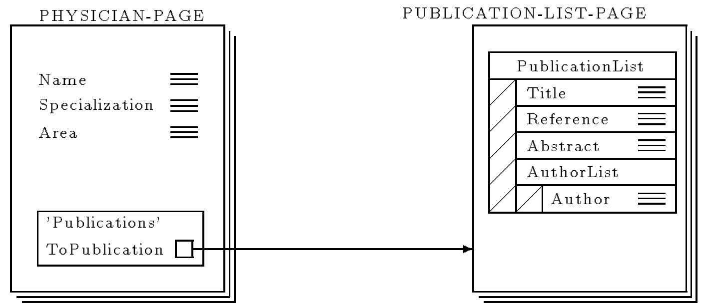



Figure 5.11: Partitioning Lists

### Horizontal Partitioning and Forms

Forms allow the selection of a horizontal partition to be implemented in a rather involved way. As we discussed in 3.1.3 forms can be seen as virtual list of links: by an effective design of its attributes, a form allows designers to express in a flexible way the selection to be associated with links.

Also, note that forms allow users to select a specific subset of the target list with incomplete information (for example using regular expressions to fill some fields), and, if useful, to specify the ordering criterion of the resulting list set.




PHYSICIAN-PAGE.Name $\in$ PUBLICATION-PAGE.PublicationList.AutorList

Figure 5.12: List `PublicationList` is horizontally partitioned

We have presented the restructuring transformations, as well as the conditions that may suggest their application, on the basis of qualitative considerations. However, it is straightforward to extend our discussion in order to allow the designer to specify definitive conditions to decide if a certain transformation is needed. For example, using estimates on the extension of data, such as the cardinalities of each macroentity and of each list, the designer can monitor the average number of items in a given list, and decide to substitute the list introducing a multilevel list or a form.

# Bibliography


[1] IBM NetData Home Page. `http://www.ibm.com/netdata`.

[2] Informix Home Page. `http://www.informix.com`.

[3] The Oncolink Web site. `http://www.oncolink.upenn.edu`.

[4] Oracle Home Page. `http://www.oracle.com`.

[5] S. Abiteboul and R. Hull. IFO: A formal semantics database model. *ACM Transactions on Database Systems*, 12(4):297–314, December 1987.

[6] S. Abiteboul and R. Hull. Data functions, Datalog and negation. In *ACM SIGMOD International Conf. on Management of Data*, pages 143–153, 1988.

[7] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[8] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. `http://www-db.stanford.edu`, 1996.

[9] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. *Journal of Digital Libraries*, 1(1):68–88, April 1997.

[10] A. Albano, R. Bergamini, G. Ghelli, and R. Orsini. An object data model with roles. In *International Conf. on Very Large Data Bases (VLDB'93), Dublin*, pages 39–51, 1993.

[11] P. Atzeni, G. Mecca, and P. Merialdo. To Weave the Web. In *International Conf. on Very Large Data Bases (VLDB'97), Athens, Greece, August 26-29*, pages 206–215, 1997. `http://poincare.dia.uniroma3.-it:8080/Araneus/`.

[12] P. Atzeni and D.S. Parker Jr. Formal properties of net-based knowledge representation schemes. In *Second IEEE International Conference on Data Engineering, Los Angeles*, pages 700–706, 1986.

[13] P. Atzeni and D.S. Parker Jr. Set contaiment inference. In *First International Conference on Data Base Theory (ICDT'86), Roma, Lecture Notes in Computer Science 243*, pages 73–90. Springer-Verlag, 1986.

[14] C. Batini, S. Ceri, and S. B. Navathe. *Conceptual Database Design: an Entity-Relationship Approach.* Benjamin and Cummings Publ. Co., Menlo Park, California, 1993.

[15] C. Batini, G. Di Battista, and G. Santucci. Structuring primitives for a dictionary of entity relationship data schemas. *IEEE Transocation on Software Engineering*, 14(4):344–365, 1993.

[16] P. Buneman. Semistructured data. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'97), Tucson, Arizona*, pages 111–121, 1997.

[17] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'96), Montreal, Canada*, pages 505–516, 1996.

[18] T. Catarci, L. Iocchi, D. Nardi, and G. Santucci. Conceptual views over the Web. In *Proceedings of the Fourth Workshop on Knowledge Representation meets Databases (KRDB'97) (in conjunction with VLDB'97)* `http://sunsite.informatik.rwth-aachen.de/Publications/-CEUR-WS/Vol-8/`, 1997.

[19] R. G. G. Cattel. *The Object Database Standard ODMG-93.* Morgan Kaufmann Publishers, San Francisco, CA, 1994.

[20] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogenous information sources. In *IPSJ Conference, Tokyo*, 1994.

[21] P.P. Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.

[22] S. Cluet, C. Delobel, J. Simon, and S. Smaga. Your mediators need data conversion! In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'98), Seattle, Washington*, 1998. To Appear.

[23] E. Damiani and L. Tanca. Semantic approaches to structuring and querying Web sites. In *IFIP Working Conference on Database Semantics*, 1997.

[24] R.A. ElMasri and S.B. Navathe. *Fundamentals of Database Systems*. Benjamin and Cummings Publ. Co., Menlo Park, California, 1988.

[25] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. STRUDEL – a Web site management system. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'97), Tucson, Arizona*, 1997. Exhibits Program.

[26] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the boat with STRUDEL: Experiences with a Web–Site management system. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'98), Seattle, Washington*, 1998. To Appear.

[27] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A query language and processor for a Web site management system. In *Proceedings of the Workshop on the Management of Semistructured Data (in conjunction with ACM SIGMOD 1997)* `http://www.research.att.com/~suciu/-workshop-papers.html`, 1997.

[28] P. Fraternali and P. Paolini. A conceptual model and a tool environment for developing more scalable, dynamic, and customizable Web applications. In *VI Intl. Conference on Extending Database Technology (EDBT'98), Valencia, Spain, March 23-27*, 1998.

[29] F. Garzotto, L. Mainetti, and P. Paolini. Hypermedia design, analysis and evaluation issues. *Communications of the ACM*, 58(8):74–86, August 1995.

[30] F. Garzotto, P. Paolini, and D. Schwabe. HDM – a model-based approach to hypertext application design. *ACM Transactions on Information Systems*, 11(1):1–26, January 1993.

[31] G. Gottlob, M. Sherry, and B. Rock. Extending object-oriented systems with roles. *ACM Transactions on Information Systems*, 14(3):268–296, July 1996.

[32] R. Hull. A survey of theoretical research on typed complex database objects. In J. Paredaens, editor, *Databases*, pages 193–256. Academic Press, 1988.

[33] R. Hull and M. Yoshikawa. ILOG: Declarative creation and manipulation of object identifiers. In *Sixteenth International Conference on Very Large Data Bases, Brisbane (VLDB'90)*, pages 455–468, 1990.

[34] R.B. Hull and R. King. Semantic database modelling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[35] R.B. Hull and C.K. Yap. The format model: A theory of database organization. *Journal of the ACM*, 31(3):518–537, July 1984.

[36] T. Isakowitz, E. A. Stohr, and P. Balasubramanian. RMM: A methodology for structured hypermedia design. *Communications of the ACM*, 58(8):34–44, August 1995. `http://www.stern.nyu.edu/~tisakowi/rmm/`.

[37] M. Lenzerini. Covering and disjointness constraints in type networks. In *Third IEEE International Conference on Data Engineering, Los Angeles*, 1987.

[38] G. Mecca, A. Mendelzon, and P. Merialdo. Efficient queries over Web views. In *VI Intl. Conference on Extending Database Technology (EDBT'98), Valencia, Spain, March 23-27*, 1998. To Appear.

[39] F. Paradis and A. M. Vercoustre. A language for publishing virtual documents on the Web. In *Proceedings of the Workshop on the Web and Databases (WebDB'98) (in conjunction with EDBT'98)* `http://-poincare.dia.uniroma3.it:8080/webdb98`, 1998.

[40] B. Pernici. Objects with roles. In *International Conference on Office Information Systems*, pages 205–215. ACM, 1990.

[41] G. Poncia and B. Pernici. A methodology for the design of distributed Web systems. In *Conference on Advanced Information Systems Engineering (CAiSE'97)*, 1997.

[42] D. Schwabe and G. Rossi. The Object-Oriented Hypermedia Design Model. *Communications of the ACM*, 58(8):45–46, August 1995.

[43] G. Simeon and S. Cluet. Using YAT to build a Web server. In *Proceedings of the Workshop on the Web and Databases (WebDB'98) (in conjunction with EDBT'98)* `http://poincare.dia.uniroma3.it:8080/webdb98`, 1998.

[44] K. Takahashi and E. Liang. Analysis and design of Web-based information systems. In *Sixth International World Wide Web Conference (WWW'97)*, 97. `http://www6conf.slac.stanford.edu/`.

[45] T.J. Teorey and J.P. Fry. *Database Design*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[46] D. Tsichritzis and F.H. Lochovski. *Data Models*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[47] A. M. Vercoustre and B. Dell'Oro, J. ans Hills. Reuse of information through virtual documents. In *Second Australian Document Computing Symposium*, pages 53–64, Melbourne, Autralia, April 1997.

[48] A. M. Vercoustre and Paradis F. A descriptive language for information object reuse through virtual documents. In *Fourth Int. Conf. on Object Oriented Information systems (OOIS97)*, pages 229–331, Brisbane, Autralia, November 1997.

[49] M. Winslett. Databases and the World Wide Web, 1997. Tutorial presented at ICDE'97.