Maintenance of Data and Metadata in Web Site Management Systems

Giuseppe Sindoni
Giuseppe Sindoni

Maintenance of Data and Metadata in Web Site Management Systems

Thesis Committee
Prof. Paolo Atzeni (Advisor)

Reviewers
Dr. Anne Marie Vercoustre
Dr. Karl Aberer
Preface

Metadata, i.e. collections of information about the structure, the meaning and the use of system data, are assuming an ever increasing importance in next generation information systems [100, 51]. The high degree of distribution and heterogeneity of most application domains and in particular the advent of the World Wide Web [35] have made metadata a crucial feature of many new kind of systems. One of the most meaningful examples are Health Information Systems, where many different divisions of hospitals and other institutions are involved and sophisticated data-sharing facilities are needed [65]. These new applications have led to the introduction of metadata models, standards and management systems in different areas [75, 54, 4]. Many proposals however, face the metadata management problem from a specific, domain-dependent perspective and the proposed solutions are not always applicable in other areas. The need of providing applications with an uniform access to Web data, has also started many efforts at defining standard models for describing the content and the structure of a Web document [7]. Unfortunately, these proposals start either from a vision of the Web as a whole document database, or of a single Web document as a database, without considering the structures that may be possibly present in a Web site. As Web sites are evolving from pure data sink or sources towards more structured information repositories, we believe in the importance of a different approach, where each Web site is seen as a database, with its logical structure, which should be described by means of metadata. Moreover, these metadata should be made available to any external application in order to access the site and extract the needed information. For example, a Web site that publishes medical records, may be accessed either by physicians, who are mainly interested in the single pieces of information, or by researchers, who may be interested in extracting aggregate data in order to give values to indicators.

The Web context has a number of interesting challenges in the development of metadata models and systems. For instance, metadata should be modeled and deployed on the Web in a complete standard way, because any external application should be allowed to extract information from the site, independently from the computing platform of both the server and the client. In addition, as many large sites rely on a database management system from which data are dynamically extracted and embedded in pages, a problem arises with metadata. The generated pages are in fact normally virtual: they are not stored neither in the server nor in the client, they disappear as soon as the client application closes the connection. This approach does not allow querying applications for accessing metadata about the structure of the site. In fact, if the pages are dynamically generated only after a specific request, in order to access all the site metadata, the application should issue all the possible database queries, which is impossible in most cases. A different approach is hence needed, where it is possible to make up-to-date metadata available on-line. It should be possible to adopt a materialized approach for data-intensive Web pages. Pages should be made persistent by storing them in the server and by maintaining them
consistent with updates in the underlying database.

To address this problem, we propose an approach to the publishing and maintenance of data and metadata in the Web that has been developed in the framework of the Araneus project [1].

More specifically, we will describe a system for publishing meta-information about the logical structure of a site [33]. It is based on a specific logical model for describing the scheme of a Web site and on walG (the Weaving ALGebra [102]): a nested algebra with invention of identifiers. The algebra, based on the scheme, describes the correspondences between the data storage model and the presented hypertext. A Data Definition Language called Penelope Definition Language (PDL) implements the algebra and allows for the definition of mappings between an underlying relational database and either HTML sites with embedded metadata or XML sites. Finally, a Data Manipulation Language called Penelope Manipulation Language (PML) [85] allows for the generation and update of the pages of the site according to the defined site-database mappings.

To the best of our knowledge, most current research prototypes and commercial systems for publishing databases on the Web do not explicitly allow for the persistence of pages. Thus, they do not provide site developers with powerful tools for the automatic generation and publishing of metadata about the structure of the site. In this thesis a strong rationale for a materialized approach to Web site generation is discussed and an implemented solution is proposed. In particular, we believe our tools contribute in bringing to the Web the same metadata maintenance philosophy that is used in databases catalogs: the same model and language are used to manage both site data and metadata.

An algorithm to guarantee consistency between the current states of both structured and semistructured data [102] has been defined. The algorithm uses a specifically designed data structure that maintains information about the relationships between the structured and semistructured data managed by the system, i.e. between the database tables from which the published data and metadata come and the site pages through which data and metadata are presented.

The maintenance of a materialized hypertext can be related to the one of maintaining materialized views in both relational and object-oriented databases. This problem has been widely debated by database researchers. However, our algorithm applies to hypertext that contain data coming from a relational database. As it will be illustrated in the follow, we model a hypertext according to an object-based model. We then introduce materialized object views over relational databases, which is a rather new concept, quite important in the Web, where there is a strong need of both publishing legacy data and developing from scratch new, data-intensive sites. The solution to the maintenance problem proposed in this thesis extends hence previous ideas in the area of materialized view maintenance to hypertext views over relational databases.

The thesis is structured as follows.

Chapter 1 introduces metadata and Web-based information systems in the context of Health Information Management.

In Chapter 2 an implemented prototype system, called MRBRAQUE (Medical Report Browsing And Querying), will be described. It enables the efficient management of collections of multimedia documents and it represents a good application
example in which the above problems come into evidence. Here, the main project issues have been: (i) to exploit the textual component of a multimedia document to give a more evident semantic meaning to the non-textual data and (ii) to use a semantic network of allowed terms to describe and bind the structure and contents of the hypermedia reports. The experiences made in developing the MRBRAQUE system have been a good starting point for further research activities aimed at the definition of general-purpose models and tools for Web-based applications.

Chapter 3 describes a general architecture for Web-based information systems. ADM, a data model for Web sites which allows the description of the logical structure of hypermedia documents, will be introduced, together with WALG: the object algebra that allows to describe hypertext views over structured data.

In Chapter 4 it will be shown how the model and language may be efficiently used to embed meta-information into the pages of an hypertext and how the approach is compatible with XML, the recently proposed standard markup language on which future Web sites will most likely be based.

When structured data are updated, consistency must be maintained between these data and the corresponding hypertext presentation. In Chapter 5 the connected problems will be described and an auxiliary data structure and an algorithm to guarantee such consistency will be presented.

Chapter 6 contains an overview of the related literature in the fields of metadata, WBISs, heterogeneous data integration, management of medical documentation and maintenance of materialized views.

Finally, in Chapter 7 some concluding remarks are discussed and some open problems are presented.
Contents

1 Introduction and Motivation ............................................. 1
  1.1 Metadata ......................................................... 3
  1.2 Web Based Information Systems ............................... 5
    1.2.1 Problems of Web Based Information Systems ........... 6

2 The MRBRAQUE System .................................................. 9
  2.1 A Model for Multimedia Medical Documents ................. 11
    2.1.1 Managing Medical Reports ............................. 11
    2.1.2 Metadata for Multimedia ............................... 12
    2.1.3 Metadata for Semantic Constraints .................... 14
    2.1.4 A Metadata-Centred Hypermedia Model ............... 16
  2.2 The System Architecture ....................................... 19
  2.3 Towards General Models and Tools for the Web ............. 20

  3.1 The Components of a Web-base Management System .......... 22
  3.2 The ARANEUS Data Model .................................... 25
  3.3 The Weaving Algebra ......................................... 25
  3.4 Site-Database Mapping with PENEOLE ....................... 28

4 Web Metadata Generation .............................................. 31
  4.1 Embedding Meta-Information into the Structure ............ 32
  4.2 The eXtensible Markup Language ............................. 33
  4.3 Generation of XML Sites ..................................... 38

5 Incremental Maintenance of Web Data and Metadata .......... 43
  5.1 Rationale for Materializing Pages ............................ 44
    5.1.1 The Problem ............................................. 45
    5.1.2 Contributions .......................................... 46
  5.2 Maintaining a Materialized Hypertext ....................... 46
    5.2.1 A Manipulation Language for Web Hypertext .......... 46
    5.2.2 Incremental Page Maintenance ........................ 47
  5.3 Automatic and Incremental Page Maintenance ............... 50
    5.3.1 Design Requirements ................................... 50
    5.3.2 The Page Scheme Dependency Graph .................... 51
    5.3.3 An Algorithm for the Maintenance of Page Scheme Instances .... 52
6 Overview of Related Works  
6.1 Metadata in Health Information Systems . . . . . . . . . . . . . . . . . . . . 58  
6.2 Metadata for System Integration . . . . . . . . . . . . . . . . . . . . . . . . . 58  
6.3 Metadata for Web Databases . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59  
   6.3.1 A Classification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59  
   6.3.2 Standards . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60  
   6.3.3 Metadata for Web Site Design . . . . . . . . . . . . . . . . . . . . . . . . . . 60  
6.4 Materialized Hypertext Maintenance . . . . . . . . . . . . . . . . . . . . . . . 61  
   6.4.1 Hypertext View Generation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61  
   6.4.2 Incremental Maintenance of Materialized Views . . . . . . . . . . . . . . . 61  
7 Conclusions and Open Problems  
7.1 Development of Web–based Medical Applications . . . . . . . . . . . . . . . 64  
7.2 Publishing of Web Metadata . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65  
7.3 Maintenance of Materialized Web Sites . . . . . . . . . . . . . . . . . . . . . . . 66
List of Figures

2.1 Enhancing the Semantics of Multimedia Documents with Metadata. . 13
2.2 The Object Model of a Thesaurus for Medical Reports. ............. 15
2.3 Hypermedia Model for a Collection of Medical Reports. .......... 17
2.4 The MRBRAQUE Prototype Architecture. ......................... 19

3.1 A General Architecture for Web Based Information Systems. ....... 22
3.2 The Araneus WBMS. ............................................. 24
3.3 A portion of a Hospital Division web scheme. .................... 26

4.1 A sample HTML source generated by Penelope with embedded meta-
    tags ................................................................. 34
4.2 The DTD file corresponding to the ClinicianPage page scheme .... 39
4.3 A sample XML source, whose DTD is depicted in Fig. 4.2 .......... 41

5.1 A Portion of the Page Scheme Dependency Graph for the Hospital
    Department WEB Scheme. ...................................... 52
Chapter 1

Introduction and Motivation
The main distinctive features of a health care system are *distribution* and *heterogeneity* [52, 65].

The patient care process, in fact, normally involves many different divisions of a hospital, where data are collected about the patient, the examinations that have been performed on the patient, the patient’s care progresses, etc. This implies that data about the overall patient’s care history are normally scattered into more than one archive that are distributed across the whole hospital, if not across more than one hospital. Moreover, these archives are very often characterized by a certain degree of heterogeneity, due to differences in data formats, media, logical schemes and data semantics.

If we look at the evolution of health care practices from a broader perspective, we are assisting the development of new forms of health care enterprises: regional health care networks are now tying together patients, primary care practitioners, community hospitals, tertiary medical centers, and a variety of other supporting services. To become efficient, certain services such as radiology and pathology have begun to develop teleradiology and telepathology systems, further loosening geographic and institutional constraints. Multi-institution health care chains are now incorporating geographically remote services. The health care system is moving from a scenario made of individual providers towards a federation where the former are components of new enterprises. Health care is becoming a cooperative activity in which patients, physicians, institutions and resources must share information in order to improve their effectiveness. Many of the future health care activities will be carried on through interactive networks: between a patient at home and a doctor in office, among a team of doctors in consultation using an on-line decision aid, etc.

In such a scenario, Health Information Systems (HISs) may be an effective support to patients’ care, if they satisfy the following requirements.

**Ubiquity**

Applications should be portable or accessible from different locations: a practitioner needs to be able to move from place to place or cover patient care while at home as well at an office without being forced to change work habits. This implies also that applications should be accessible via a variety of different hardware and software platforms.

**Integrated interfaces**

Consistent user interfaces should be developed so that applications integrating various functions do not require a different form of interaction for each function. These interfaces must also be *complete*: that is, they must make available all the capabilities of an HIS within the framework of a single application, avoiding duplicate entry of information to accomplish related tasks.

**User-friendliness**

Different categories of users should be allowed to interact differently with the system. Patients will have different cultural levels, ability to type, availability of hardware, etc. Professionals may require portable devices with flexible data entry tools.

Some users prefer viewing information in different formats, e.g. graphical vs.
narrative vs. tabular. Some require adaptability to both portable and desktop use, depending on location. Depending on the kind of user, different views into the same information resources and presentation formats are most appropriate.

Applications should also support different points of view on the available information to match the tasks of different users. We may for example have a chronological, speciality–based, body system–oriented, problem–oriented or management–oriented view on the same collection of data.

**Extensibility**

The system should be able to incorporate new requirements deriving from new emerging needs without excessive modifications. In the new globally distributed scenario such changes will be quite frequent, so a high degree of flexibility will be necessary for keeping the system useful for a long time.

We believe that the satisfaction of these requirements may be achieved by stressing the following design paradigms:

- developing *metadata-centred information systems*;

We therefore illustrate the two concepts in general and the perspective from which we deal with them in this thesis.

### 1.1 Metadata

Given a data management system, its metadata are a collection of data about the structure, the meaning and the use of the system data.

Metadata may be completely disconnected from the actual data (e.g. metadata about images or videos), and may be possibly created by third parties. They are usually more about the content of information rather than its structure.

When dealing with the *structure* of information, metadata are normally managed by means of the same model and languages of the system in such a way to be straightforwardly used by applications.

This thesis is mainly focusing on metadata about the structure of information, in the spirit of the database community. In particular, we try to extend to the Web context the strategy of using the database schema as metadata, like for example in relational catalogues.

The concept of metadata is in some sense orthogonal to many database and information systems research areas. We devised the following as the most explored ones.

**Metadata for Digital Media.** (See for example [49, 67, 75, 88, 95, 96].) Here, metadata are used to enhance the semantics of multimedia data and queries. Metadata systems are built to cope with the intrinsic heterogeneity of data coming from different media. These systems are used to model, under a common logical structure,
the descriptions of each multimedia component. Metadata abstract information content independently from the location, format or media of data and enable correlation of information at a semantic level across multiple forms and representations.

The main problems here include: devising general models and languages for metadata design and access; coping with the impossibility of exact match in comparing values; easing the analysis process of huge data sets; balancing content based and attribute based access to data to minimize the former for application and performance reasons; and finally emphasizing the importance of derived data, context and semantics representation, which are necessary when dealing with audio–visual data.

**Metadata for Heterogeneous Information Systems Integration.** (See for example [54, 99, 98].) The vast area of system integration has produced a number of efforts aimed at defining metadata approaches to the detection and resolution of conflicts among system semantics. The general methodology foresees the definition of metadata models to represent data semantics. Metadata may be explicitly stored or defined by means of environments. The overall aim is to build context mediators, that is, software modules able to perform semantics reconciliation tasks among heterogeneous systems.

Outstanding problems are: defining a suitable framework to detect class and attribute similarity; developing metadata–based languages and tools for values restructuring and composition; and devising methods for the automatic extraction of metadata.

**Metadata for Digital Libraries.** (See for example [4, 103, 109].) Researches in this area are mainly aimed at supporting the modeling of digital documents and user queries in the context of catalogue production for vast collections. Records must be created that are more informative than an index entry but less complete than a formal cataloguing record. The descriptions should follow established standards in order to ease the development of automated tools for metadata collection.

Dublin Core [4], for example supports very external descriptions of Web pages, image databases and digital libraries in general, and does not attempt to describe their internal structure.

The main challenges in the area include [26]: defining appropriate representations for objects not amenable to digitization; tuning the quality of the provided information objects with respect to the variability in users’ tolerance; providing consumers with on–line ontologies and classification systems to help them retrieve information and locate expected and desired resources; using metadata to ease the query expansion and refinement processes; finally, providing mechanisms for securing information and auditing access.

**Web Metadata.** (See for example [33, 40, 56, 106, 7].) The Web is characterized by heterogeneity and loose structure. The notion of semistructured data [20] is a very well known one in the Digital Libraries and Document Management communities and applies quite well to the Web context. The problems deriving from these features of the managed data types have originated some efforts aimed at using a meta–data approach to their resolution. In particular the main goals have been so far: to define models and languages to embed metadata into semistructured documents, in order to
make them more easily accessible by querying applications; to define ontologies for the sharing of documents of heterogeneous type and structure among different classes of users; to define meta–models to be used for the integration of information coming from the different data sources that contribute to the information content of a site; to propose meta–data centred approaches to the design process of data-intensive web sites. Some of these activities are currently being analyzed by the W3C groups in order to propose related metadata standards. In particular, the eXtensible Markup Language (XML [16]) and Resource Description Framework (RDF [13]) standards are likely to provide a better Information Retrieval support by the use of database-like query languages.

Most of the problems and challenges concerning metadata in other fields, apply also to the Web context. Some currently interesting topics include: developing meta-applications able to access, in a declarative fashion, a collection of semistructured documents without any knowledge of their structure and content; extending the metadata approach to the modeling and design of the dynamic aspects of a WBIS, like site update applications, and, more generally, Web-based Computer Supported Cooperative Work applications; and investigation on quality aspects of metadata.

In particular, as many Web sites are indeed large on-line repositories of semistructured data, we believe that the time is mature to start talking of Web Based Information Systems.

1.2 Web Based Information Systems

Web Based Information Systems are systems for the management of multimedia data in distributed environments that rely on both database and Internet-related technologies. These systems are able to effectively manage, in an integrated fashion, both highly structured data, using one or more DBMSs, and semistructured Web-style data, using those technologies that are becoming ever more popular with the growth of the World Wide Web (WWW) [35]: markup languages\footnote{like the HyperText Markup Language (HTML) [64] or the more recently proposed eXtensible Markup Language (XML) [16]}, HyperText Transaction Protocol (http) servers [83] and platform-independent programming languages (like Java [57]).

In a WBIS one or more DBMSs are usually used for storing and updating data, which are presented to users in an integrated fashion as hypermedia documents. Hypermedia documents in the Web are in essence collections of HTML or XML files and are delivered to users by an http server. Pages may be generated dynamically, on demand, or may be materialized, as it will be clarified in the follow. The files are normally created by querying the database to extract the needed data according to a suitable model for the hypermedia ([27, 34, 53, 61, 63, 72, 85, 90, 92, 94, 101]). The hypermedia can then be accessed by any internet-enabled machine running a Web browser. Platform–independent applications for data querying, manipulation and update may be developed as CGIs or Java applets.

Clearly, a WBIS may be very useful in order to solve some of the problems we addressed for Health Information Systems because:
• the hypermedia paradigm allows the definition of an abstract logical structure for data presentation to the user;
• a different structure may be defined to present the same data in a different manner to different classes of users;
• standard, platform-independent, markup languages are used to encode the hypertext\(^2\) files that are sent across the network; access to data is then possible from any computer, operating system and browser;
• the flexibility of models for semistructured data allows for an easy extension of the system to meet new requirements.

1.2.1 Problems of Web Based Information Systems

The nature of WBISs implies a number of problems that are related to the distribution of the applications among different sites and users, the availability of data at different degrees of structure and the need of providing access and updating functions to both structured and semistructured data. We devised the following as some of the most important:

1. **A uniform interface for heterogeneous data types.**

   Medical documentation is inherently heterogeneous. The Patient Record (PR) [18], for example, is a document containing patient’s personal data, which are normally stored in a DBMS; results of exams, which may consist of multimedia data with text annotations; a physician’s observations about the patient’s health status; drugs prescriptions, etc. Despite this high degree of heterogeneity, access to medical documentation must be provided in a uniform fashion to all the actors in the care process. Efficient query languages and browsing functions must be devised in order to ease the production and use of documents by physicians, nurses, administrators, etc. This implies the necessity of a general framework to describe, store and query all the needed information about the structure and semantics of the various component of the document.

2. **Publishing of logical structures.**

   There cannot be made any a-priori assumptions about the client applications that access the documents. This implies that a paradigm must be devised to embed a description of the logical structure of documents into the presented documents themselves. The physical media used to store and present documents must to encapsulate enough information about the structure, the meaning and the use of the presented multimedia data.

3. **Maintenance of materialized files.**

   As will be illustrated in the sequel, the publishing of metadata suggests, as a valid option in many contexts, the materialization of hypertext, i.e. the pages of the hypertext need to be actually stored in the site server. Suitable auxiliary

\(^2\)Throughout the rest of the thesis the terms “hypertext” and “hypermedia” will be used indifferently to mean a collection of linked documents.
data structures and algorithms must be designed to guarantee the consistency between the structures and contents of the single heterogeneous components and those of the integrated presentation.

The remainder of the thesis illustrates a research path that starts from the design and development of a metadata-centred medical report management system based on Web technologies. This experience led to the idea of defining a general model and a number of tools for Web Site Management Systems Development.

We will first present a system prototype for the management of medical reports. The system is called MRBrAQue and is based on (i) a metadata base describing the features of each kind of media [88], (ii) a vocabulary in which each kind of document is semantically described by explicitly storing the names and structures of its sections, all the terms that are allowed into each section and the relationships between terms [49], (iii) the use of an object oriented data model and language for describing the structure of both data and metadata.

From the experience made with MRBrAQue we learned that, in order to ease the development of effective, user-friendly and long-lasting Web sites, a specific model for the logical structure of the site and a language for site database mapping is needed. This model and language have been defined in the framework of the ARANEUS project [1], a project aimed at developing new models and tools for Web applications. In particular, a solution to the problem of metadata publishing and consequent site maintenance are introduced in the follow.
Chapter 2

The MRBRÁQUE System
In this section the MRBrAQe system for the management of medical documentation will be described. The system represents the result of a research activity in which a number of interesting problems in the field of metadata for medical information systems have been investigated and some ad-hoc solutions have been proposed. These solutions have been a good starting point for the study and the definition of the more general models and tools that will be presented in the sequel of the thesis.

The overall goal of the MRBrAQe project is to investigate the problem of effectively creating, retrieving and browsing multimedia medical documents, to define novel methods and tools and to develop a prototype based on the devised principles. In this framework, documents are virtual, i.e. they are dynamically generated by retrieving the component multimedia data. These data, which will be referred in the sequel as underlying data, are separately stored in different databases.

To achieve a real semantic integration of multimedia data, the focus of attention is on the modeling approach, which is metadata-centered, and on the modalities of interactive exploration of the document collections, which are based on both the browsing and querying paradigm. The system allows for the retrieval of information by (i) manually browsing the hypermedia network through a HTML-based interface and (ii) dynamically defining queries to drive the navigation towards subsets of documents that satisfy certain requirements. This idea that has been proposed before in the field of hypermedia [3]. Such proposals rely mainly to a querying paradigm that is based on Information Retrieval techniques. Partial and similarity matching are allowed and the user is presented with a set of links that point to relevant documents. Our approach is instead based on an object-oriented query language that allows only for exact matching between query values and attribute values of the target document collection. We lose the flexibility of partial matching, but the user is sure that the retrieved documents are exactly those that meet the information needs. This is reasonable in a system for managing the medical reports of a hospital division, because in such an environment having a complete control on and a standardization of the used terminology is one of the main requirements.

As a practical experimentation of these concepts, a system (MRBrAQe: Medical Report BRowsing And Querying) has been implemented. It allows users to compose the report according to a protocol that is defined and driven by a simple knowledge base (KB). The KB represents at the same time the semantics of the specific reporting context and the structure of the report, following a design pattern similar to the one presented in hypermedia modelling [17].

The MRBrAQe system shares, with other research prototypes\(^1\), the basic idea of annotating multimedia data, like still images, with metadata. These systems are mostly aimed at using the semantics of information inherent into images in order to treat them as database items. We inspired from these approaches to extend the idea of annotating images to a situation in which textual data do not exist only as an indexing support for images, but they have their own dignity as first class citizens of the database.

In the rest of the chapter the proposed multimedia modeling approach is presented: the emphasis is on the importance of the use of metadata for enhancing the data semantics and, consequently, the semantics of the expressible queries. The overall

\(^1\)Like for example VIMSYS [70]. The early VIMSYS prototypes have evolved into a commercial system, called Virage [14]
2.1 A Model for Multimedia Medical Documents

In our approach, a multimedia medical document is a complex object composed of several parts, each using a different medium to represent its information content.

We exploit the structure of the textual component of a multimedia document to give more semantics to the non-textual ones, in order to solve incompleteness and clearness problems. To date, the system allows for documents composed by text, images and structured data, that is, data directly managed by a DBMS. The underlying idea is that to achieve an effective management of pictorial data, they have to be mapped into structures that represent their semantics and the structures must find a correspondence in the descriptive part of the document.

In the sequel, the main issues of the medical reporting activity will be briefly illustrated. Then, the concept of metadata as information used to describe the heterogeneous type of data of a multimedia document will be presented. In particular, we will illustrate how a terminological system can be used to represent the concepts of the application domain together with their relationships and, at the same time, to enforce a structure on the textual component of the document. Finally, we will present our model for medical hypermedia.

2.1.1 Managing Medical Reports

The medical report (MR) represents one of the most important kinds of document in the patient care process. It is used by the physicians to choose therapeutic strategies and it represents a useful comparison means to evaluate the progress of a disease, a post-surgical state, or the effectiveness of a therapy. Besides, a large enough report collection, if supplied with proper consultation tools, can be a useful information repository from which physicians can extract interesting relationships among the various kinds of available information and discover facts that are hidden into the unstructured nature of the report.

Most of the diagnostic examinations are based on images: the radiologist is responsible to inspect them in order to answer the specific questions put by the physicians. Relevant image contents are described and, if it is possible, a diagnosis is performed. Then the report is stored in some repository with its images and included in the patient folder: any authorized health operator can read it.

It is not unusual that reports are not sufficiently complete and clear [89]. For example, descriptions are not enough detailed and terminology can be ambiguous or incorrect. Besides, it is not efficient to archive the report and the related multimedia data with traditional paper based documents and slides: the relationships between descriptions and multimedia data are not directly maintained. These relationships hide in fact most of the radiologist’s knowledge and it is then worth to make them available to the system users.

The application of our methods provides physicians with tools for creating and organizing large collections of hypermedia reports. Hybermedia reports are documents in which administrative and personal data, numeric data coming from exam results, texts, images and graphics are presented in an integrated fashion and links are defined.
among the components to enrich their semantics and ease the browsing and querying of the report collection.

2.1.2 Metadata for Multimedia

In a health information system, the same multimedia object is often accessed and manipulated for different purposes by different applications and, since it is usually better to avoid replicas, giving unique semantics to multimedia is not very useful. To give multimedia objects different semantics we need to describe their media types and their contents with respect to different applications and domains [88]. Meta-information modeling, in our perspective, is the key to achieve this goal. This meta-information, as it will be shown in the follow, is described, stored and accessed using the same models and language that are used for the documents data.

In this section, we show how the meta-information is used for defining correct semantics to multimedia objects with respect to their accessibility and manageability by distributed applications having different aims. In particular, we present an example that explains how, to an increasing use of metadata, more semantics for data and queries correspond.

We consider documents which are composed by text and still images. It is of course possible to extend a document with components having others media types, such as video, graphics, animation, sound, etc. Moreover, the single components need not to be stored in the same site, but we can have a distributed multimedia environment in which Web documents can be created taking components from different sites. The semantics of a multimedia document regards the meaning of each component with respect to the document. The more semantics we can express, the more we can increase the querying and browsing capabilities of applications.

Fig. 2.1 shows three schematic solutions for the management of a collection of distributed multimedia documents. We start with a situation in which the only kind of available information is location and format of each component and nothing is known about the relationships among components. The second solution is enriched by information describing relationships among components seen as black-boxes. The semantically richest solution foresee information about relationships among parts of the various components. We now illustrate into detail the three situations.

In the lower situation, text component and image component are stored in flat files. To allow querying about the document as a whole, meta–information about document composition (e.g. URL and media type of each component) and metadata about the storage of the whole document (e.g. last modification date of the last modified component) are needed. Moreover, media-related metadata (e.g. format and compression technique of images) must be provided in order to allow document presentation. This information allows to query the system for both single components of the document and the document as a whole. For instance, it may be possible to formulate queries like the following.

Retrieve image components from reports dated after 31-12-95.

This is of course a very poor system, but it is important to take it into account because it emphasizes how the capabilities of the query language depends on the metadata it can manage: the raw data of the image component and the ASCII data
of the text component do not contribute to satisfy the query. If users are interested in richer queries, this solution is not satisfactory.

One step beyond can be done adding more metadata to the document model. In particular, systems should be able to support querying on component content; for instance, a radiologist who queries a report collection may be interested in retrieving images where a given *remark* has been observed or images concerning a given *anatomical structure*. This can be achieved only by adding information describing image contents. In the same manner we consider keywords on the text component, if we consider them as text content descriptors, i.e. metadata about text. Moreover, as some of the keywords can refer to parts of the image components, if also information about these relationships is managed, such keywords may be used to define query conditions on the content of the single components of a document. For instance, in the text component keywords may be found that are remarks or anatomical structures: they are both metadata about the text and the image component. Such a model allows for queries like the following.

Retrieves images where an ulcer has been remarked.
In the upper part of Fig. 2.1 a structure is depicted where a new type of metadata allows to solve the problems existing with the previous ones. Here, metadata are used to describe those parts of an image which terms refer to and to maintain information about relationships among parts of images and terms. Query like the following will be satisfied.

Retrieve parts of images where an ulcer has been remarked.

In order to achieve the goal, the system manages Regions Of Interest (ROI). A ROI is a zone of an image which locates some interesting feature. In a Web page, it is possible to define anchors on ROIs into images, allowing to attach links to ROIs. The concept of ROI can be easily extended to other types of multimedia objects: ROI descriptors depend on media type. For instance, in ECG (ElectroCardioGraph) tracings, a ROI is represented by the start and the end points that delimit a portion of interest in the tracing.

2.1.3 Metadata for Semantic Constraints

We allow the use of metadata for validation, through schema and constraints, of each document component. A set of terms, meaningful for the specific domain and organized according a semantic network, is used to guide the composition of the textual component of the report and the construction of links among document components. The network, like the data it describes, has been modeled using an object-oriented (OO) methodology [93] and is managed by an OO DBMS. Following a physicians’ habit, we call it on-line controlled thesaurus, but it may be viewed as a simple ontology or knowledge base.

The Model of the On-line Controlled Thesaurus

The logical model for the thesaurus is depicted in Fig. 2.2.

The central class of our report model is the EXAM TYPE class. Its instances are all the names of the examinations that are currently allowed in the reporting system. Each class of the model has a unique attribute because an instance of a class is an object, with a unique attribute, representing a medical term. For example, an instance whose attribute value is the string EsophagoGastroScopy represents one of the possible examinations for which it is possible to define a report.

The model describes a general structure for medical reports as a set of relationships among classes. We have in fact a number of classes representing useful information for the different types of examinations. For example we may have different therapeutic procedure types for each examination, that is, possible therapies that may be applied to the patient during the examination. They are represented by the THERAPEUTIC PROCEDURE TYPE class, each instance of which is the name of a possible therapeutic procedure.

It is worth to put into evidence that the results of an examination are structured into descriptive sections, represented by the DESCRIPTIVE SECTION TYPE class. Each descriptive section is essentially a set of possible remarks. A remark is anything noticeable that the physician who performed the examination has observed on the exam results.
The knowledge base is maintained by one of the physicians of the division, who is in charge of:

- defining all the allowed terms (i.e., all the exam types, all the conduction types, etc.) for the report composition activity of the division;
- defining the instances of the relationships between classes, which implicitly define the allowed instances of reports;
- inserting new terms and instances of relationships, modifying existing terms and deleting existing terms and relationships.

The associations play then a very important role for the documents management: they represent the constraints for retrieving the collection of new terms and associations that is possible to insert into the report in a given moment of the composition process, as it will be clarified in the follow.

Finally, the schema of the knowledge base reflects the schema of the textual sections of the report. That means that an instance of report, as far as its textual part is concerned, is the result of instantiation of metadata coming from the knowledge base.
Metadata-driven Report Composition

The report composition activity consists of two phases:

1. choosing, from the knowledge base, the terms that must be used to instantiate all the needed report sections;

2. defining relationships between terms and images coming from the examination.

In this section we will focus on the dynamic interaction between the user and the knowledge base, while in the follow we will present the hypermedia model and the functions for associating terms with regions of images.

In MRBrAQe, the textual part of a report consists of general information about the examination, remarks on the results and diagnostic conclusions. General information and remarks are structured into classes and relationships that reflect the structure of the knowledge base. Text is in fact only generated as the result of metadata instantiation. Diagnostic conclusions instead, are directly edited by physicians as free text and they are included in the report as a specific section.

During the document composition process, the system accesses the semantic network in a context-sensitive way, and presents the user with only the subset of terms that are allowed in that specific context. For example, if EsophagoGastroScopy has been chosen as the name of the exam and the user wants to insert a THERAPEUTIC PROCEDURE TYPE, the system extracts from the knowledge base all the instances of THERAPEUTIC PROCEDURE TYPE that are associated with the EsophagoGastroScopy instance of EXAM TYPE (say Diathermal coagulation and Polypectomy) and presents them as a list from which the desired term can be chosen and inserted into the report instance.

Each moment of the writing activity is then characterized by a terminological context, i.e. a subset of the semantic network. In this way, the thesaurus plays the role of common terminological reference for all those who are involved in the document production activity.

2.1.4 A Metadata-Centred Hypermedia Model

As we said above, and as it has been illustrated in literature [36, 59] a hypermedia document model meets the need of storing and presenting the relationships among multimedia data that characterize medical documents: the hypermedia links allow to describe, store and present most of the knowledge that the user embeds into the textual part of the document. Our hypermedia management system stores multimedia nodes, anchors and links using the same physical model and provides users with browsing, querying and updating capabilities on both the system data and metadata.

We model a multimedia document in such a way to impose a suitable structure to its textual part. By suitable structure we mean that the information contained in the report must be conceptually modeled in order to enforce a high degree of standardization in the reporting activity of the division. Moreover, this structure is automatically

---

2In general, a report may or may not contain all the types of information that are represented in our model.
2.1. A MODEL FOR MULTIMEDIA MEDICAL DOCUMENTS

derived from the thesaurus that stores all the relevant domain concepts. This allows also for terminological control, preventing different physicians to describe the same phenomenon with different words. During the composition of the textual sections of the report, choosing a term means choosing, for the section we are composing, the structure that is described by the relationships the term has with other terms of the thesaurus. Finally, in order to provide the report with full hypermedia features, each term in the textual part of the document may be linked with one or more ROIs in the associated images.

The proposed model for the medical report consists of five main components [49] and is illustrated in 2.3.

1. Patient data.
2. Examination description.
3. Examination products (e.g. images).
4. Semiotic description.
5. Diagnosis.

Figure 2.3: Hypermedia Model for a Collection of Medical Reports.

This structure partly reflects the meta-model represented by the thesaurus model of Fig. 2.2. In particular, the "Semiotic description" component maps to the DESRIPTIVE SECTION TYPE class and to its directly and indirectly associated classes (REMARK TYPE,
MORPHOLOGICAL CHARACTERIZATION TYPE, QUALITATIVE CHARACTERIZATION TYPE and LOCATION TYPE) and the "Examination description" component maps to all the classes that are associated to EXAM TYPE.

The patient data component models information about the patient’s characteristics (SSN, sex, age, etc.) and they come from an existing hospital database. This information allows the system to create inter-document links to browse for example all the reports concerning a patient.

Exam descriptions represent the diagnostic protocol chosen by the diagnostician: data about exam conduction, used contrast means, etc.

The semiotic description consists of a list of clinical remarks, each of which associated with topological, morphological and qualitative characterizations. For example a part of a semiotic description may be the following.

Presence, in the post-pylorus, near the lower wall, of a deep ulcer lesion, 0.5 cm wide.

The following table describes how this sentence is modeled in our framework.

<table>
<thead>
<tr>
<th>REMARK</th>
<th>TOPOLOGIC CHAR.</th>
<th>MORPHOLOGIC CHAR.</th>
<th>QUALITATIVE CHAR.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ulcer lesion</td>
<td>in the pos–pylorus, near the lower wall</td>
<td>0.5 cm. wide</td>
<td>deep</td>
</tr>
</tbody>
</table>

Each remark and topological characterization into this section of the report may be an anchor to originate a link to anchors on the multimedia data the radiologist is describing. These links represent intra-document relationships and allow the user to immediately correlate the semiotic descriptions with the ROIs they refer on the images. With these anchors we may construct more complex inter-document relationships. For example, reports containing the same remarks but referring to different patients may be dynamically linked. The anchors on images are regions that are represented by simple graphic primitives (rectangles, circles, arcs, segments) and are drawn by the user during the document compiling phase.

The lower layer represents the documents organized in two collections: the descriptive components of each document and the multimedia products of exams. Links between pairs of elements of the two collections represent the ROI-semiotic description relationships and must to be explicitly stored by the system. Links between descriptive sections represent relationships among documents. This kind of link is dynamically built up by queries over structured data. For example, if a query is issued to retrieve all the documents relative to the same patient, the system dynamically links together the resulting set and presents it to the user as an indexed tour of Web pages.

Inter-layer links are useful to create another kind of inter-document relationship. For example, we may have links between two semiotics descriptions, and related ROI, coming from two or more different reports, if they refer to the same type of remark. Here, the metadata structure allows for browsing the report collection according to condition on terms of the medical domain. Essentially, when a term appears in a report, say ulcer, we know that it is in the knowledge base and that each term of the
2.2. The System Architecture

In Fig. 2.4 the high level architecture of the prototype is depicted.

We will focus our attention on the conceptual interface, which implements most of the run time support for the novel functions of the system. It manages in fact the interaction between the OODBMS and the graphical interfaces of the system. It is divided into two distinct subsystems:

1. the report creation subsystem,
2. the report consultation subsystem.

The report creation interface implements the functions for hypermedia document producing and archiving. The report consultation interface implements the interaction functions between the consultation gui and the hypermedia collection. They are composed essentially by a number of CGIs and they allowed the development of the user interfaces as dynamic standard Web sites.
2.3 Towards General Models and Tools for the Web

The experience of the development of the MRBRAQUE prototype led us to a new and more general research project aimed at generalizing some of the solutions that have been proposed in the medical context. In fact, when we tried to extend the approach to other contexts, we realized immediately that it has the following limitations.

- The absence of a specific logical model for Web sites forced us to develop ad-hoc user interfaces, hard-wiring into HTML code the CGIs for querying and updating the network of documents.

- The virtual approach to hypermedia document composition, while optimizing the use of space, prevents the access to the documents from other Internet applications. A different approach, allowing for the generation of persistent Web pages from the database data may allow applications based on Web query languages (like for example the ones proposed in [23, 28, 29, 80]) to access the document collection for different purposes.

- The CGI–based procedural approach to the development of query applications makes their maintenance a very cumbersome one. A procedural language for defining mappings between data and Web pages may result in a powerful tool for site generation and maintenance.

In the sequel, we will present some proposals that arose in the framework of the Araneus project [1], whose aims are the development of models and tools for the efficient management of large Web sites.
Chapter 3

Web–based Information Systems: Models and Tools
The general architecture of a Web-based information system is represented in Figure 3.1. In our perspective, a WBIS must be based on a Web-Base Management System (WBMS) [85]. In this chapter, we will illustrate the concept of WBMS, describe the components and the main features of a WBIS and present ADM, a reference data model for Web sites, and WALG, an algebra for describing mappings between structured and semistructured data [102], which is one of the original contributions of this thesis. Finally, we will describe the Araneus approach to Web data and metadata generation and how it is suitable for both HTML and XML sites.

![Diagram of a General Architecture for Web Based Information Systems](image)

Figure 3.1: A General Architecture for Web Based Information Systems.

### 3.1 The Components of a Web-base Management System

As is shown by Fig. 3.1, in a WBIS both databases and Web sites may be either local or remote resources. The user interacts with the system through the Internet, hence a standard interface should be developed in order to provide platform independence.
The interface is normally written in HTML, but future systems seem likely to be based on the new XML proposal. Two classes of Web applications are supported by a WBIS: update applications and query applications.

Update applications allow for the maintenance of the site. Update pages constitute a site in itself and can be accessed by a specific group of users who are allowed to add, modify and delete site data.

Query applications allow access to and browsing of the heterogeneous data: suitable high-level languages are needed for both structured and semistructured data [24, 28, 50]. Moreover, data coming from heterogeneous sources should possibly be restructured and integrated in views, the access to which is again possible through the query user interface.

In such a scenario, changes in the outside world, i.e. database and Web sources updates, should be reflected on the semi-structured views.

All the above functions for a WBIS are supported by the Web-Base Management System, which plays with respect to a WBIS the same role played by a DBMS with respect to a classic IS.

The work presented in this thesis has been conducted in the framework of the Araneus project [1], whose aim is the development of the Araneus WBMS [85]. Its overall architecture is depicted in Fig. 3.2.

The system uses the ADM data model, an ODMG-like model for describing a Web hypertext. ADM, which will be described in the next section, allows the system for the management of nested object by decomposition. The DBMS is in fact used to store the objects in flat tables.

The hypertext query module allows the system to extract data from Web-sites that may be either remote or local. The system makes use of Ulixes [29], a suitable language based on a navigational algebra. The sites may be either external, i.e. remote, or local. As will be clarified in the sequel, with respect to local sites the query process may be made simpler by embedding metadata in the site pages. These metadata describe the logical structure of the site pages and can be directly accessed by query applications in order to extract data without the need to write specific wrappers [50, 85], as must be done for external sites, whose structure is not a priori known to the system.

The creation and management of new sites is performed by the Penelope module. Data to be published are stored in a database and the module allows for the generation of a new hypertext whose logical structure has been previously defined using ADM. As will be illustrated in more details in the sequel, the system allows for two approaches to page generation: the common approach by which pages are dynamically generated as the result of a query on the database, and a different approach by which pages are stored in HTML files and “pushed” to the site. There are of course advantages and disadvantages to both approaches that will be discussed in Chap. 5. It is here worth noting that in general a dynamic approach to page generation does not allow any system for metadata publishing, since pages are virtual and they exist only after a query has been issued. This means that only a few metadata on the structure are generally available through the model of the embedding page and no metadata about the content is externally available to query systems [74], and that an indexing system
Figure 3.2: The Araneus WBMS.
3.2 THE ARANEUS DATA MODEL

Aiming at creating such a metadata catalogue would have to issue all the possible database queries in order to perform its task, which is in most cases impossible and anyway potentially expensive.

The Penelope module is based on WALG, the weaving algebra, which can be used to define ADM-based hypertext views over database tables. WALG has been defined as the basis of an implemented language called PDL (Penelope Definition Language [31]), which allows for the declarative definition of a hypertext. Hypertext maintenance is performed by PML (Penelope Manipulation Language [102]).

In the following we will describe ADM, which is the reference logical model for the contributions presented in this thesis, WALG and Penelope.

3.2 The Araneus Data Model

The reference data model that we adopt for hypertext views is a subset of the Araneus Data Model, and it will be described in the following with the help of the scheme of Fig. 3.3, which gives a graphical intuition of how the model can be used to model the logical structure of a hypertext.

ADM is a page-oriented model, because page is the main concept. Each hypertext page is seen as an object having an identifier (its URL) and a number of attributes. The structure of a page is abstracted by its page scheme and each page is an instance of a page scheme. The notion of page scheme may be assimilated to that of relation scheme, in the relational data model, or object class, in object oriented databases. In Fig. 3.3 for example, home pages of patients are described by the PatientPage page scheme.

Each PatientPage instance has three simple attributes (PatName, RecoveryDate and Sex). Pages can also have complex attributes: lists, possibly nested at an arbitrary level, and links to other pages. The example shows the ClinicianPage page scheme, having a list attribute (PatientList). Its elements are pairs, formed by a link to an instance of PatientPage and the corresponding anchor (PatName), and a nested list (Exams) of exams performed on the patient. The reader interested in the details of the full model, may see [31].

It is worth putting into evidence that to each page scheme a template is associated, which contains all graphical page presentation details. It is thus possible to separate the description of the structure of a page from its presentation, in the spirit of XML [16].

3.3 The Weaving Algebra

WALG, the weaving algebra, allows for the description of a mapping between a relational database and a hypertext view according to the ADM model. WALG is essentially a nested relational algebra extended with an URL invention operator, which allows an identifier manipulation mechanism to create complex hypertext structures, in a manner similar to the one presented by the authors of [71].

The operators of the algebra work on nested relations and return nested relations, as follows:
• *nest* \([46]\), \(\nu_{A \leftarrow B}\), is an operator that groups together tuples which agree on all the attributes that are not in a given set of attributes, say \(B\). It forms a single tuple, for every such group, which has a new attribute, say \(A\), in place of \(B\), whose value is the set of all the \(B\) values of the tuples being grouped together;

• *generate* \(URL\), \(URL_{I \leftarrow K}\), is an operator that adds a new attribute to a (nested) relation. The value of \(I\) for each tuple of \(R\) is uniquely determined by the values of the attributes in \(K\) in the tuple. Thus, it is a Skolem function of these values. The system interprets this values as URLs. If the name of the generated attribute is URL, in \(URL_{URL \leftarrow K}\), it means that the system uses its value as a page URL for the (nested) tuple. In any other case, the value of \(I\) is interpreted as an URL that already exists, thus is a link from this nested tuple (viewed as a page) to another page. \(I\) serves as the anchor of the link.

More formally:

![Figure 3.3: A portion of a Hospital Division WEB scheme.](image-url)
3.3. THE WEAVING ALGEBRA

\[ \text{URL}_{I \rightarrow K}(R) = \{ R'(I, K, X) \mid t[I] = OID(K) \land t[K, X] \in R \} \]

- other relational operators are extended to the nested relational model, for example in the way illustrated in [46].

Let us have for example the following database, storing data that are presented in the above site (key attributes are underlined).

\[
\begin{align*}
\text{PATIENTS} & (\text{PatName}, \text{Sex}, \text{RecoveryDate}, \text{PhysName}); \\
\text{PHYSICIANS} & (\text{PhysName}, \text{Speciality}, \text{Type}, \text{Seniority}); \\
\text{EXAMS} & (\text{EName}, \text{Date}, \text{PatName}, \text{Result}, \text{ValueName}, \text{VUnit}, \text{Value}, \text{VMax}, \text{VMin});
\end{align*}
\]

The WALG expression corresponding to the ClinicianPage view, i.e. to all the ClinicianPage instances (pages), is the following \(^1\).

```walg
ClinicianPage = \\
\nu \text{PatientList} \leftarrow (\text{PatName}, \text{ToPatient}, \text{Exams}) \\
\nu \text{Exams} \leftarrow \text{EName}, \text{Date}, \text{ToExam} \\
\text{URL} \leftarrow \text{PhysName}, \text{ToPatient} \leftarrow \text{PatName}, \text{ToExam} \leftarrow (\text{EName}, \text{Date}, \text{PatName}) \\
\Pi \text{PhysName}, \text{Speciality}, \text{Seniority}, \text{PatName}, \text{EName}, \text{Date} \\
\sigma \text{Type} = '\text{Clinician}' \\
(\text{PHYSICIANS} \bowtie \text{PATIENTS} \bowtie \text{EXAMS})
```

Here, the three base tables PHYSICIANS, PATIENTS and EXAMS are joined to give a resulting relation having the following scheme.

\[
R(\text{PhysName}, \text{Speciality}, \text{Type}, \text{Seniority}, \text{PatName}, \text{Sex}, \text{RecoveryDate}, \text{EName}, \text{Date}, \text{Result}, \text{ValueName}, \text{VUnit}, \text{Value}, \text{VMax}, \text{VMin})
\]

A selection is performed on the tuples of this relation, to retain only the tuples relative to clinicians.

Three new attributes are then added. Here PHYSICIANS.\text{PhysName}, PATIENTS.\text{PatName} are the URL base attributes – i.e. the database attributes whose values are the arguments of the OID generation function – for the URL and ToPatient attributes respectively; while the instances of the ToExam URL attribute are generated from the values of three attributes of the EXAMS table: EName, Date and PatName. The invention mechanism guarantees the integrity of the generated hypertext: for example the values of the ToPatient attribute and the corresponding URLs of the instances of the PatientPage page scheme are generated by the application of a Skolem function to the same base values.

Before performing the nesting operations it is necessary to get rid, by a projection operation, of all the unnecessary attributes, i.e. all those attributes that are not mapped to the ClinicianPage page scheme. The resulting relation has the following scheme:

\(^1\)We assume that the Type attribute is used to distinguish the various types of physicians in the hospital.
A nesting is then performed on the \textit{EName}, \textit{Date} and \textit{ToExam} attributes, to form a nested relation with a new nested attribute (\textit{Exams}). The final nesting is performed on the \textit{Exams} attribute itself, to give a second level nesting, and on the \textit{PatName} and \textit{ToPatient} attributes. The final structure of the nested relation reflects the \texttt{ClinicianPage} page scheme of Fig. 3.3.

The weaving algebra is an abstraction of an implemented language, called \texttt{Penelope} [31, 85], which allows the definition of the structure of a Web site according to its ADM model.

In the next section we will briefly introduce \texttt{Penelope} and discuss its data publishing features.

### 3.4 Site-Database Mapping with \texttt{Penelope}

The \texttt{Penelope} language supports both materialized and virtual solutions for page generation: pages can be either generated from the database content and materialized on a server, or they can be dynamically delivered to the browser after a user's explicit request. The advantages of materializing database-derived sites will be discussed in Chap. 5. While in Chap. 4 it will be presented the ability to automatically embed meta-information in HTML code, which allows pages to be queried using suitable tools.

In \texttt{Penelope}, page structure is described by \texttt{DEFINE PAGE} statements: they essentially specify how to generate pages based on database table attributes. For example, the following \texttt{Penelope} \texttt{DEFINE PAGE} statement generates the HTML code for the page instances corresponding to the \texttt{ClinicianPage} page scheme of Fig. 3.3:

\begin{verbatim}
DEFINE PAGE ClinicianPage
AS URL URL(<PhysName>);
   PhysName: TEXT <PhysName>;
   Speciality: TEXT <Speciality>;
   Seniority: TEXT <Seniority>;
   PatientList: LIST OF
      (PatName: TEXT <PATIENTS.PatName>;
      ToPatient: LINK-TO URL(<PATIENTS.PatName>))
   Exams: LIST OF
      (EName: TEXT <EName>;
      Date: TEXT <Date>;
      ToExam: LINK-TO
         URL(<EName>,<Date>,<EXAMS.PatName>))
FROM PHYSICIANS, PATIENTS, EXAMS
WHERE Type = 'Clinician'
IN DIVDB
\end{verbatim}
The name of the underlying database (DIVDB) is specified in the IN clause, while the FROM clause indicates the database table(s) or view(s) that contains relevant data. A left join is performed on these tables by the Penelope interpreter. The AS clause specifies the filling-out of data in pages. For each page, a different URL is created by the use of function terms. Filling-out of pages is specified by attribute definitions. For example, the values of the PhysName TEXT attribute come from attribute PhysName of the PHYSICIAN database table. Finally, coherently with the ADM page scheme structure, the definition of the PatientList attribute specifies how, for each patient, a link to the corresponding page must be established. This is accomplished by using as an anchor the patient name and by using as a value for the the link reference the function term URL(<PATIENTS.PatName>).

The Penelope source code must of course be enriched by proper tagging directives, in order to instruct the interpreter about the desired page layout.

It is finally worth putting into evidence that the Penelope interpreter can be used for the generation of both materialized and dynamic Web pages. A specific tagging directive may in fact be added to a LINK TO attribute in order to instruct the interpreter that the target page must be dynamically generated whenever the link is activated. Generation is performed by means of a CGI that extract from the database the needed information and generate the target page according to the scheme of the site.

In the next chapter, we will present the metadata publishing features of Penelope, with a special emphasis on the publishing of XML pages.
Chapter 4

Web Metadata Generation
Many limitations in the query capabilities of existing Web information extraction systems derive from the fact that the HTML standard doesn’t offer any explicit framework for document metadata management. Starting from this limitation and aiming at clearly splitting the definition of the layout from that of the structure of a Web document, the World Wide Web consortium now recommends XML [16]. The eXtensible Markup Language is a syntax for defining new and context-dependent markup systems that allow for the explicit description of the structure of a document. It seems, therefore, the ideal candidate as the reference document format for providing Web-based cooperative applications with the classes of functions illustrated in Sec. 3.1 [39]. Unfortunately, the database perspective has not been taken into account during the XML definition process and the corresponding proposal lacks the notion of site scheme and specific features to describe site-database mappings: consequently, XML meta-information framework as it stands, cannot tell anything about the relationships between structured and semistructured data. It therefore needs to be effectively supported by suitable models and tools for designing mappings between databases and XML sites. The aim of this section is therefore to present XML and to discuss its relationships with the Araneus approach to the generation of database-derived Web meta-information. We will show how this approach can be effectively used for both HTML- and XML-based systems. At the moment, metadata management in Araneus is performed by the Penelope Definition Language [31], that allows the descriptions, in a declarative fashion, of the mappings between site pages and database data, according to the ADM model for the site. The adopted logical modeling approach enables the Penelope interpreter to automatically generate the site pages (data) and to embed, into standard HTML comments, suitable meta-tags (metadata), which describe the page structure and its data content.

Finally, the relationships between ADM and XML will be described and it will be shown how our approach has been properly extended to the automatic generation of XML sites. Generated sites are therefore: (i) more suitable for declarative structured querying by navigational languages and advanced query systems (see for example [31, 82]); (ii) self-described, because the implicit knowledge contained in the logical scheme of the site is made explicit into XML declarations and tags; and (iii) compliant with a recommended and widely accepted standard and therefore suitable for data exchange in cooperative applications.

4.1 Embedding Meta-Information into the Structure

It is very useful to keep track of page structures in the generated HTML files in such a way as to make this meta-information available to Web-based applications. This is accomplished by embedding meta-tags into HTML comments, thus making them completely transparent to ordinary Web browsers; they can however be used by more sophisticated applications in order to extract relevant pieces of information from the page.

Consider for example the HTML source shown in Fig. 4.1. Some hidden tags have been added, in addition to actual data to be displayed. The first of these tags, in the page header <!-- PAGE-SCHEME ClinicianPage ... -->, describes the structure
of the page-scheme according to which the page is organized. Then, for each attribute in the page, the corresponding value is marked by suitable meta-tags in order to easily recognize and extract the value.

Meta-tags generation is enabled by setting an environment variable. The PENELOPE interpreter generates the ADM page schemes and embeds them into the page instances, together with the proper metadata of each single data value.

Once the page has been organized in this way, applications can be developed that make use of page data. For example it may be possible to query the site, i.e. to automatically navigate the site to extract information based on high-level queries; this is often desirable when accessing large amounts of data, in order to avoid the usual disorientation associated with browsing. Ulixes [31, 30] is a tool that has been explicitly designed to this end; queries over a site can be expressed based on the corresponding ADM scheme using simple path-expressions, in order to access pages and store data in a local database. As an alternative, straightforward extensions of W3QS [79] or WebSQL [87] could be used, or even a combination of an HTTP robot and a grammar parser. In this way, the resulting site is not just a bunch of HTML files, but a highly structured repository that can be either browsed or queried, thus making data access more effective. More sophisticated applications may be built on top of such query tools, in order to customize data use according to specific requirements.

The embedding mechanism for meta-tags presents however some major drawbacks. In particular:

- it does not allow for explicit separation between comments (which are supposed to be pieces of information about the HTML code itself and not about data structure) and metadata (which are pieces of information about the structure and content of the document to be used by applications);

- applications must be tailored for a specific embedding syntax;

- it essentially doubles page sizes, making site management heavy;

- in order to separate document layout and structure descriptions, specific application modules must be implemented.

For example, in order to overcome the latter, we have developed a specific tool that allows the separate management of the definition of a page scheme logical structure, which is stored in templates, and the definition of its presentation, which is described in stylesheets.

The ability to generate XML files for describing page storage and structure may solve those problems.

4.2 The eXtensible Markup Language

The aim of this section is to give some basic notions about the main features of XML, in order to enable the reader to understand the concepts illustrated in the following sections. For more details, see [41, 60, 108, 16].
Figure 4.1: A sample HTML source generated by Penelope with embedded metadata
XML is a “meta-markup” language and it allows the definition of tagging systems to explicitly represent the structure of a document as a tree of nested objects. A piece of XML document may in fact look like this:

```xml
...<Clinician>
    <PhysName>Fabrice Grooms</PhysName>
    <Speciality>Surgery</Speciality>
    <Seniority>15</Seniority>
    <PatientList>
        <PatName>Mark Eater</PatName>
        ...
    </PatientList>
...</Clinician>...
```

XML documents are composed by markup and content. In the above example, the two tags `<PhysName>` and `</PhysName>` are the markup for the string Fabrice Grooms, which is content.

**Elements.** There are different kinds of markup that can appear in an XML document. In particular, the example presents element markups. Elements identify the nature of the content they surround. Each element begins with a start-tag and ends with an end-tag.

**Attributes.** Another important kind of markup is attributes. Attributes are name-value pairs that occur inside tags after the element name. For example, `<PatName identifier="ME">` is the PatName element with the attribute identifier having the value ME.

**Document Type Declarations.** Document type declarations syntactically define the tagged structure of a document, allowing at the same time the presentation of meta-information about its content. Declarations are used by application parsers to validate documents. In fact, they define the allowed sequences and nesting of tags, attribute values and their types and defaults, the names of external files and the formats of some external (non-XML) data that may be included. The most important kinds of XML type declarations are element declaration and attribute declaration.

**Element Declarations.** These identify the name of elements and the nature of their content. For example, the following declaration identifies the element named Clinician.

```xml
<!ELEMENT Clinician (PhysName, Speciality, Seniority, PatientList)>  
```

This element must contain exactly one occurrence of the element PhysName, followed by exactly one occurrence of the element Speciality, followed by exactly one occurrence of the element Seniority, followed by exactly one occurrence of the element PatientList. Declarations for all the sub-elements must also be present for an
XML processing application to check the validity of a document. For example, the definition for PatientList might be the following:

```xml
<!ELEMENT PatientList (PatName+)>  
<!ELEMENT PatName(#PCDATA)>  
```

PatientList must contain at least one occurrence of Patient. Patient occurrences are #PCDATA, which is a keyword that means “Parseable Character DATA” and is reserved to indicate character data.

**Attribute Declarations.** They identify element attributes. For example, we may have the following attribute declaration.

```xml
<!ATTLIST PatName  
    identifier ID #required>  
```

Here, the element PatName has an attribute called identifier, it is of type ID (a specific type that means essentially an identifying name) and the presence of the corresponding value is compulsory.

Attribute declarations are used essentially to:

- define the set of attributes pertaining to a given element type;
- establish type constraints for these attributes;
- provide default values for attributes.

**Validating and non-Validating Applications.** The content of an XML document might in principle be processed without a type declaration. The start-end tagging mechanism is in fact enough for a parser to recognize the object tree and the nesting structure. However, there are contexts where a rationale exists for requiring a declaration. For example, most authoring environments need to read and process document type declarations in order to understand and enforce the content models of the document. With respect to our specific application environment, even if the structure of the generated document is guaranteed by adm and Penelope, if we want XML documents to be parsed and queried by specific tools, we need to make the document definitions available together with the documents themselves.

The document type declaration must be the first thing in the document and identifies the root element of the document. Additional declarations may come from an external definition file\(^1\), be included directly in the document or both. A validating application may need to read one or both the declarations, in order to perform its validating tasks.

An XML document is valid if (i) it obeys the syntax of XML and (ii) it contains a proper document type declaration and obeys the constraints of that declaration.

**Linking.** The XML linking specification (XML Linking Language, XLink [15]) is currently under development, consequently, in the following only a survey of its basic and more consolidated features will be presented.

\(^1\)The external definition is called Document Type Definition, DTD, in the spirit of SGML [105].
A link describes a relationship between any locations that are addressed in it. The nature of this relationship depends on both the processing application and the semantic information supplied by the document authoring application.

XML does not have a fixed set of elements, hence the element name cannot be used by a parser to locate links. A specific attribute named XML-LINK is then used to identify links. Other attributes can be used to provide additional information to the processor. There are essentially four kinds of links in XLink, but the most important for our aims are Simple Links.

**Simple Links.** These are quite similar to HTML links:

```xml
<ToPatient XML-LINK="SIMPLE" HREF="/PatientPage/markeater.xml">
  <PatName>John Doe</PatName>
</ToPatient>
```

They define a link between two resources: the content of the linking element itself (the `PatName` element) and another resource, which may be for example a URL or a query.

**XML and Presentation.** Since XML documents have no fixed tag set, the hard-coded approach of HTML browsers will not work for presentation. XML document presentation must therefore be based on stylesheets. There is also an ongoing effort by the W3C in this regard, the proposal is called *eXtensible Style Language* (XSL [12]) and is likely to be focused on the definition of a standard stylesheet language. Other stylesheet languages, like Cascading Style Sheets (CSS [2]) are also likely to be supported.

**XML and Web Site Design.** It may be argued that XML could be directly used as a logical model for Web site design. There are two main drawbacks in this respect:

1. XML lacks the explicit notion of site scheme, which is particularly important in the Web site design process [32].

2. In XML it is not possible to explicitly represent some classes of constraints. For example, in the current version it is not possible to bind the target object type of a link object to a given document type. The definition of link semantics is somehow supported by the attribute declaration mechanism, but, in the end, it is left to the authoring and processing applications. It has however been claimed that XLink will solve this problem.

To clarify the latter, let us have the following XML type declaration.

```xml
<?XML version="1.0" rmd="internal"?>
<!DOCTYPE PatientPage [ 
  <!ELEMENT PatName (#PCDATA)> 
  ...]>
The first line is the XML markup declaration. The markup rmd="internal" states that only the internal declarations need to be processed in order to validate the document. The second line is the internal declaration of the root element of a document. But, as the scope of the name PatientPage is limited to the document itself, the corresponding type declaration is unknown to other documents. They cannot bind the target of link attributes in the document type declarations to the PatientPage type. A way out may be to spot an external DTD file by means of a specific link attribute, and to instruct the processing application to parse the file in order to bind the link target to the declared type. In the next section this will be clarified with an example.

As a consequence of what has been illustrated so far, we think that the use of a more structured model like ADM for site scheme description and of a meta-markup language like XML for page and metadata storing, together with the Penelope language for describing database-site mapping, will enable the design, realization and maintenance of Web Base Management Systems that rely on standard and widespread technologies and that are suitable for easy integration into cooperative applications.

In the sequel, the relationships between ADM and XML declarations will be illustrated, in order to put into evidence the analogy between an ADM page scheme and an XML DTD and to show how it is possible to generate XML sites with the Araneus tools.

### 4.3 Generation of XML Sites

The Araneus Web Base Management System allows to generate HTML views over relational databases, as illustrated in the previous sections, and to query existing sites to produce relational views over an HTML hypertext [31]. However, as has been pointed out above, there are some drawbacks connected with the use of HTML as the markup syntax for the generated pages. In particular, it is not possible to separate page structure from its layout directives and the only way to export metadata about document contents is by embedding them into comments. By using XML as the syntax for document storage structure definition, a more efficient and standardized management of metadata can be achieved. Hence, in order to extend both Araneus site generation and querying to XML documents it is first necessary to analyse the relationships between ADM and the XML document type declaration syntax.

Document type declarations in XML can be internal or external. However, as has been illustrated in Sec. 4.2, internal declaration scope spans just the document in which they are declared. So, in order to correctly bind link attributes to their target type in such a way as to represent link type consistency, we must take into account external declarations (i.e. DTD files) only. In particular, we must devise a standard framework to express the fact that a given link expresses a relationship between classes of documents, because XML syntax allows for the explicit definition only of relationships between document instances. Moreover, having a unique external DTD that defines a class of XML documents (i.e. a page scheme), allows a validating application to access only one file in order to understand the structure of all the instances of the page scheme. Take for example the ClinicianPage page scheme of
4.3. GENERATION OF XML SITES

Sec. 3.2: it may be mapped on the XML DTD of Fig. 4.2.

```xml
<!ELEMENT ClinicianPage (PhysName, Speciality, Seniority, Patient+)>  
<!ELEMENT PhysName (#PCDATA)>  
<!ELEMENT Speciality (#PCDATA)>  
<!ELEMENT Seniority (#PCDATA)>  
<!ELEMENT Patient (ToPatient,Exam+)>  
<!ELEMENT ToPatient (PatName)>  
<!ELEMENT PatName (#PCDATA)>  
<!ELEMENT Exam (ToExam)>  
<!ELEMENT ToExam (EName, Date)>  
<!ELEMENT EName (#PCDATA)>  
<!ELEMENT Date (#PCDATA)>  

<!ATTLIST ToPatient  
  XML-LINK CDATA #FIXED "SIMPLE"  
  TRG-TYPE CDATA #FIXED "/DTD/PatientPage.dtd"  
  HREF CDATA #REQUIRED >  

<!ATTLIST ToExam  
  XML-LINK CDATA #FIXED "SIMPLE"  
  TRG-TYPE CDATA #FIXED "/DTD/ExamPage.dtd"  
  HREF CDATA #REQUIRED >
```

Figure 4.2: The DTD file corresponding to the ClinicianPage page scheme

Here, the mechanism to map the ToPatient and ToExam link attributes is worth noting. Each ADM link attribute is mapped into an XML element. The TRG-TYPE attribute of the link elements indicates the DTD file containing the type declarations for PatientPage and ExamPage respectively. A parsing application will then access and parse the DTD in order to extract the structure of the target document.

The general mapping rules between ADM types and XML declarations follow. The rules are used by the extended PENELlope interpreter to generate:

- the XML declarations corresponding to ADM page schemes, in such a way as to produce a consistent set of type declarations for the page schemes of the site;
- the instances of each page scheme, i.e. the XML documents.

**The ADM-XML Mapping Rules**

An ADM monovalued TEXT attribute

```xml
textatt : TEXT;
```

is mapped to the XML element

```xml
<!ELEMENT textatt (#PCDATA)>.
```
An ADM monovalued IMAGE attribute

imageatt : IMAGE;

is mapped to the XML element

```xml
<!ELEMENT imageatt (#PCDATA)>
<!ATTLIST imageatt
    XML-LINK CDATA #FIXED "SIMPLE"
    TYPE CDATA #FIXED "IMAGE"
    HREF CDATA #REQUIRED >.
```

Again, it is here worth noting the use of the TYPE attribute as a stratagem to indicate that the element is an IMAGE type.

An ADM monovalued LINK TO attribute

linkatt : LINK TO pageScheme;

is mapped to the XML element

```xml
<!ELEMENT linkatt anchor>
<!ATTLIST linkatt
    XML-LINK CDATA #FIXED "SIMPLE"
    TRG-TYPE CDATA #FIXED "pageScheme.dtd"
    HREF CDATA #REQUIRED >.
```

where anchor is the attribute which is used as the link anchor and pageScheme.dtd is the file containing the declarations for pageScheme page scheme. A processing application must then be specifically instructed that the DTD file must be parsed in order to validate the target documents.

The ADM multivalued LIST OF attribute

listatt : LIST OF (A1 : T1, A2 : T2, ...An : Tn);

where each Ai is an attribute name and each Ti is an attribute type, is mapped to the XML element

```xml
<!ELEMENT listatt (A1, A2, ... An)+>
```

and the mapping of each Ai is recursively defined.

The above mapping rules allow the PENEOPE interpreter to produce XML DTDs for the derived XML site documents.

The XML document for Fabrice Grooms’ page will then look like the one in Fig. 4.3.

The value "all" for the rmd directive means that both the internal (if present) and external declarations must be read in order to validate the document. The SYSTEM "clinician.p.dtd" directive indicates to applications the external DTD file to be
4.3. GENERATION OF XML SITES

<?XML version="1.0" rmd="all"?>
<!DOCTYPE ClinicianPage SYSTEM "clinician_p.dtd">

<PhysName>Fabrice Grooms</PhysName>
<Speciality>Surgery</Speciality>
<Seniority>15</Seniority>
<Patient>
    <ToPatient HREF="/PatientPage/markeater.html">
        <PatName>Mark Eater</PatName>
    </ToPatient>
    <Exam>
        <ToExam HREF="/ExamPage/marklapa1012.html">
            <EName>Laparoscopy</EName>
            <Date>10-12-1998</Date>
        </ToExam>
    </Exam>
    <Exam>
        <ToExam HREF="/ExamPage/markendo1112.html">
            <EName>Endoscopy</EName>
            <Date>11-12-1998</Date>
        </ToExam>
    </Exam>
    ...
</Patient>

<Patient>
    <ToPatient HREF="/PatientPage/johnbigs.html">
        <PatName>John Bigsheep</PatName>
    </ToPatient>
    ...
</Patient>

Figure 4.3: A sample XML source, whose DTD is depicted in Fig. 4.2

read. In our case it is the only document type declaration available for each class of pages.

It may be argued at this point that the ADM model can be replaced by the XML model defined by the above mapping rules. However, we must put into evidence that using a logical model like ADM provides an abstraction process on the structure of the XML repository, as it has pointed out by the authors of [86]. A clear advantage of this process is that it is possible to see the repository as a database, allowing for the separation between the logical and physical level of data.

The scheme gives a compact representation of the site content and of its semantics, supporting the formulation of queries; in fact, in the spirit of databases, query formulation based on a scheme is simpler and more accurate than working without one or with a loose one; also, the system can use the knowledge about the scheme for optimization purposes; efficiency in querying large Internet-available data sources is in fact a critical issue.

It is worth noting that querying a single document can be considered as a special case of the more general task of querying a collection of linked documents. Since
object-oriented and object-relational databases are well-established technologies, we inherit a number of consolidated solutions for querying data (extensible type system, aggregation functions, ordered and unordered collections, views and restructuring, embedded queries).

In a future work, it may be also explored the relationships between ADM and RDF [13], which has been designed for representing metadata and uses a graph model where links can have properties and be constrained.
Chapter 5

Incremental Maintenance of Web Data and Metadata
In Chap. 3 we presented a model and a language for the automatic derivation of data-intensive Web sites from a relational database.

This chapter deals with the maintenance issues required by the derived hypertext to enforce consistency between page content and database state [102]. Note that, as it has been already pointed out, all the meta-information about their structure may be embedded into the derived pages. In essence, the same model and language are use for both data and metadata, in the spirit of relational catalogs. Hence, what follows holds for both data and metadata of a data-intensive Web site.

Hypertext views are defined as nested oid-based views over the set of base relations. The ADM model is used to describe the structure of the hypertext and WALG allows to define views and view updates. A manipulation language will be presented, which allows to update the hypertext to reflect the current database state. Incremental maintenance is performed by an algorithm that takes as input a set of updates on the database and automatically produces the hypertext update instructions. Dependencies between database and hypertext are maintained by a suitable auxiliary data structure, together with logs of database updates.

The algorithm may be used for both immediate and deferred maintenance of derived data and metadata.

### 5.1 Rationale for Materializing Pages

The most common way of automatically generating a derived hypertext is mainly based on the dynamic construction of virtual pages after a client request. Usually, the request is managed by a specific program (for example a Common Gateway Interface in HTML files) or is described using a specific query language whose statements are embedded into pages. These pages are often called pull pages, because it is up to the client browser to pull out the interesting information. The pull approach unfortunately has some major drawbacks:

1. it involves a certain degree of DBMS overloading, because every time a page is requested by a client browser, a query is issued to the database in order to extract the relevant data;
2. it introduces some platform-dependence, because the embedded queries are usually written in a proprietary language and the CGIs need to be compiled on the specific platform;
3. it hampers site mirroring, because, if the site needs to be moved to another server, either the database needs to be replicated, or some network overload is introduced, due to remote queries;
4. as it has been said in Chap. 3, it doesn’t allow for the publication of some site metadata, more specifically information about the structure of the site, which may be very useful to querying applications.

Lately, a new approach is becoming more and more important among the Web operators, which is based on the concept of materialized hypertext view: a derived hypertext whose pages are actually stored by the system on a server or directly on
5.1. RATIONALE FOR MATERIALIZING PAGES

the client machine, using a mark-up language like HTML. This approach overcomes
the above disadvantages because: (i) pages are static, so the http server can work on
its own; (ii) there is no need to embed queries or script calls in the pages, standard
sites are generated; (iii) because of their standardization, sites can be mirrored more
easily, not being tied to a specific technology; finally (iv), as it has been introduced in
Chap. 3, metadata can be published either by embedding them into HTML comments
or by directly generating XML files.

The pages of a materialized hypertext are often called push pages, to remember
the fact that it is the information delivering system that generates pages and stores
them. The so-called “Web channels” (see for example [11, 8, 9]) are heavily based
on this push technology and they are assuming an ever increasing importance among
Web users.

5.1.1 The Problem

Unfortunately, some additional maintenance issues are required by a materialized
hypertext, to enforce consistency between page contents and the current database
state. In fact, every time a transaction is issued on the database, its updates must
efficiently and effectively be extended to the derived hypertext. In particular, (i)
updates must be incremental, i.e. only the hypertext pages that are dependent of
database changes must be updated and (ii) all the database updates must propagate
to the hypertext, as it will be clarified in the sequel.

A similar problem has been studied in the context of materialized relational
database views [69] and even extended to nested data models [77, 78]. However,
the heterogeneity due to the different data formats makes the hypertext view context
different and introduces some new issues. Each page is in fact stored, as a marked-up
text file, possibly on a remote server. Then, direct access to the single values into the
pages is not allowed. This means that, whenever a page needs to be updated, it needs
to be completely regenerated from the new state of the database. Moreover, con-
sistency between pages must be preserved and this is almost the same as preserving
consistency between objects.

The problem of dynamically maintaining consistency between base data and de-
rived hypertext is the hypertext view maintenance problem\textsuperscript{1}.

The are a number of related issues:

• different maintenance policies should be allowed (immediate or deferred);

• this implies the design of auxiliary data structures to keep track of database
  updates; the management of these structures overloads the system and they
  have to be as light as possible;

• finally, due to the particular network structure of a hypertext, not only has the
  consistency between the single pages and the database to be maintained, but
  also the consistency between page links.

\textsuperscript{1}The same problem has been addressed in the framework of the STRUDEL project [55] as the
problem of incremental view updates for semistructured data.
5.1.2 Contributions

This chapter deals with such issues, introducing a manipulation language for a specific class of derived hypertext; an auxiliary data structure for (i) representing the dependencies between the database tables and hypertext and (ii) logging database updates; and finally, an algorithm for automatic hypertext incremental maintenance.

Hypertext views and view updates are defined using ADM and WALG, as it has been illustrated in Chap 3.

An auxiliary data structure allows to maintain information about the dependencies between database tables and hypertext pages. It is based on the concept of view dependency graph, presented for the first time in [68], which is extended to the maintenance of the class of hypertext described by ADM.

Finally, incremental page maintenance can be performed by an algorithm that takes as input a set of changes on the database and produces a minimal set of update instructions for hypertext pages. The algorithm can be used whenever hypertext maintenance is required.

5.2 Maintaining a Materialized Hypertext

Once a derived hypertext has been designed by means of ADM and WALG, a manipulation language is of course needed to populate the site with page scheme instances and to maintain those instances when database tables are updated. The language is based on invocations of algebra expressions. Moreover, whenever a transaction is issued on the underlying database, hypertext pages become obsolete, that is, their current content does not correspond to the current database state. An extension to the system is then needed, in order to enforce, in an incremental manner, consistency between database and hypertext.

5.2.1 A Manipulation Language for Web Hypertext

In order to perform page creation and maintenance, a simple language has been defined, which is composed of two instructions: GENERATE and REMOVE. They allow to manipulate hypertext pages and they can refer to the whole hypertext, to all instances of a page scheme or to pages that satisfy a condition.

The GENERATE statement has the following general syntax:

\[
\text{GENERATE ALL | <PageScheme> [WHERE <Condition>].}
\]

The semantics of the GENERATE statement is essentially to create the proper set of pages, taking data from the base tables as specified by the page scheme definitions. The ALL keyword allows the generation of all the instances of each page scheme. PageScheme is the name of an ADM page scheme. In this chapter, we will still refer to the simple hospital site described in Sec. 3.2. So, for example, the execution of the following statement creates all the ClinicianPage page scheme instances, according to Expr. 3.1.

\[
\text{GENERATE ClinicianPage;}
\]
Finally, it is possible to generate only a subset of a page scheme instances, according to a boolean condition on the attributes of the base tables whose values are involved in the URLs generation, as it will be clarified in the sequel.

The REMOVE statement has a similar syntax:

\begin{verbatim}
REMOVE ALL | <PageScheme>
WHERE <Condition>.
\end{verbatim}

It allows to remove the specified sets of pages from the hypertext. Condition is again a boolean condition on the attributes of the base tables whose values are involved in the URLs generation.

### 5.2.2 Incremental Page Maintenance

The following example will illustrate the hypertext maintenance problem and clarify the semantics of the manipulation primitives.

Let us insert the new patient Turlow into the PATIENT table and assigning him doctor Grooms as the treating physician: the corresponding inserted tuple is [Turlow, M, 15-10-1998, Grooms]. This insertion will affect, among the others, the Clinician-Page view, because the view definition involves the PHYSICIANS table (see Expr. 3.1). The Web site pages need to be updated accordingly.

The simple “brute force” approach to the problem would simply regenerate the whole hypertext on the new state of the database, clearly performing a huge number of unnecessary page creation operations. A more sophisticated approach may regenerate only the instances of the ClinicianPage scheme, again unnecessarily regenerating all the pages relative to clinicians different from dr. Grooms. The optimal solution is hence to regenerate only the page instance corresponding to dr. Grooms.

The corresponding WALG expression follows. It describes the regeneration of the dr. Grooms clinician page. The selection condition in fact, extracts only tuples which values are presented in the dr. Grooms clinician page.

\begin{align}
\text{ClinicianPage}^{Reg} = \\
\nu \text{PatientList} \leftarrow (\text{PatName}, \text{ToPat}, \text{Exams}) \\
\nu \text{Exams} \leftarrow \text{EName}, \text{Date}, \text{ToExam} \\
\text{URL} \leftarrow \text{PhysName}, \text{ToPat} \leftarrow \text{PatName}, \text{ToExam} \leftarrow (\text{EName}, \text{Date}, \text{PatName}) \\
\pi \text{PhysName}, \text{Speciality}, \text{Seniority}, \text{PatName}, \text{EName}, \text{Date} \\
\sigma \text{Type} = \text{’Clinician’ AND PhysName} = \text{’Grooms’} \\
\text{(PHYSICIANS} \bowtie \text{PATIENTS} \bowtie \text{EXAMS})
\end{align}

(5.1)

The database transaction is extended by the following manipulation statement:

\begin{verbatim}
GENERATE ClinicianPage \\
WHERE PhysName = ’Grooms’;
\end{verbatim}

Let us now consider a scenario of deletion.

Given the relation instances\(^2\).

\(^2\)The EXAMS relation scheme has here been simplified.
The nested table corresponding to Expr. 3.1 is the following:\footnote{The \textit{Speciality} and \textit{Seniority} attributes are omitted for the sake of simplicity.}

```
<table>
<thead>
<tr>
<th>EName</th>
<th>Date</th>
<th>PatName</th>
<th>ValueName</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECG</td>
<td>3-4-1998</td>
<td>Turlow</td>
<td>BP</td>
<td>75/120</td>
</tr>
<tr>
<td>Spirometry</td>
<td>4-5-1998</td>
<td>Turlow</td>
<td>FVC</td>
<td>2.5</td>
</tr>
<tr>
<td>Spirometry</td>
<td>4-5-1998</td>
<td>Turlow</td>
<td>FEVL</td>
<td>3</td>
</tr>
<tr>
<td>ECG</td>
<td>5-5-1998</td>
<td>Mash</td>
<td>BP</td>
<td>80/115</td>
</tr>
<tr>
<td>ECG</td>
<td>5-6-1998</td>
<td>Dumbys</td>
<td>BP</td>
<td>80/120</td>
</tr>
</tbody>
</table>
```
### 5.2. MAINTAINING A MATERIALIZED HYPERTEXT

<table>
<thead>
<tr>
<th>ClinicianPage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URL</strong></td>
</tr>
<tr>
<td>http://...</td>
</tr>
<tr>
<td>http://...</td>
</tr>
<tr>
<td>http://...</td>
</tr>
</tbody>
</table>

The resulting nested relation will still contain a tuple for patient Turlow, as the database deletion involves a nested value in the hypertext. Hence, the corresponding page (that is the dr. Grooms’ page) should not be removed from the hypertext, but simply regenerated, from the new state of the base tables. The corresponding WALG expression is the same as Expr. 5.1.

The transaction is again extended by:

\[
\text{GENERATE ClinicianPage} \quad \text{WHERE} \quad \text{PhysName} = \text{'Grooms'};
\]

which is executed on the updated database, hence regenerating the page of Dr. Grooms containing the remaining Turlow’s exam only.

On the contrary, suppose dr. Mydirs moves to another hospital. We have to remove his tuple and the associations with all the patients he was treating (Dumbys and Lybrand). A transaction is then issued on the database to delete the tuple \([\text{Mydirs, Ginecology, Clinician, 20}]\) from the \text{PHYSICIANS} table and to modify the value of the \text{PhysName} attribute in the tuples corresponding to the two patients in the \text{PATIENTS} table.

As far as the two modifications are concerned, this means regenerating all the pages corresponding to the physicians who have been assigned to the patients (beside of course the patients’ pages). Assuming that the two patients have been also assigned to dr. Marquis, this corresponds again to executing the following PML statements.

\[
\text{GENERATE PatientPage} \quad \text{WHERE} \quad \text{PatName} = \text{'Dumbys'} \text{ OR PatName} = \text{'Lybrand'};
\]

\[
\text{GENERATE ClinicianPage} \quad \text{WHERE} \quad \text{PHYSICIAN.PhysName} = \text{'Marquis'};
\]

The resulting nested relation will however not contain anymore tuples for Dr. Mydirs, hence, the corresponding page should be actually removed from the hypertext:

\[
\text{REMOVE ClinicianPage} \quad \text{WHERE} \quad \text{PhysName} = \text{Mydirs}.
\]

From the \text{WHERE} condition, the corresponding \text{URL} is generated:

\[
\text{ClinicianUrl}^{\text{Rem}} = \text{URL}_{\text{URL-PhysName}} \sigma \text{PhysName='Mydirs'} (\text{PHYSICIANS})
\]
and the corresponding page is removed from the hypertext.

Essentially, each database deletion that involves values of URL attributes, corresponds to removing the pages whose URL is created from those values.

To summarize the important points about the two examples:

- addition and modification of tuples may require addition of pages or regeneration of pages;
- deletion of tuples may require deletion of pages or regeneration of pages.

The main maintenance problems in this framework, are: (i) to automatically produce the proper conditions for the \texttt{GENERATE} and \texttt{REMOVE} statements and (ii) to distinguish database updates corresponding to page removals, from database updates corresponding to page replacements. They are solved by allowing the system to log deletions from each base table and to maintain information about which base tables are involved in URL generation of each page scheme instances.

In the following, an auxiliary data structure for managing such information will be presented. The structure is used by an incremental maintenance algorithm able to produce all the statements to update a page scheme instances, starting from a set of database updates.

5.3 Automatic and Incremental Page Maintenance

The overall aim of this work is to provide a tool for relieving a Web administrator of all the run-time page maintenance problems. In particular, an algorithm and an auxiliary data structure have been designed, which are suitable to be used both for immediate maintenance and deferred maintenance.

Immediate maintenance of hypertext pages is performed when each database transaction is immediately extended with the manipulation statements to update pages that are dependent on the transaction.

When the hypertext is maintained with a deferred approach, its update operations are postponed at a later moment and possibly activated by a triggering mechanism. Deferred maintenance involves problems of site quality analysis, concurrency control and transaction overhead, which will be subjects of further research.

5.3.1 Design Requirements

The design of the maintenance algorithm and data structure has then been focused on the following main requirements:

- basing the algorithm on the (re)generation or removal of a whole nested tuple (i.e. page) as the atomic maintenance operation: database modifications are propagated to the hypertext by executing page manipulation operations, as it will be clarified in the sequel;
- designing an algorithm to asynchronously maintain the instances of a given page scheme, in such a way that it may be used either for immediately extending a database transaction on which the page scheme depends, or for deferred maintenance after a condition triggering.
• creating a core data structure to maintain essential information about nested object views and to log database updates, in order to ease distinguishing pages to be removed from pages to be regenerated and to allow deferred maintenance also.

5.3.2 The Page Scheme Dependency Graph

The system has then been designed to effectively manage the sets of inserted and deleted tuples of each base relation since the last hypertext maintenance operation. These sets will be addressed as $\Delta^+$ and $\Delta^-$. $\Delta$ have been introduced for the first time and in a different context by the authors of [62], and their nature and use in this framework will be clarified in the sequel.

The auxiliary data structure is a graph that allows to maintain information about dependencies between base relations and derived views. It is conceptually derived from the view dependency graph defined in [77], which is extended to maintain information about oid attributes, i.e. the attributes of each base relation whose values are the arguments of the oid generation operator. In Fig. 5.1, part of the dependency graph for the example site of Fig. 3.3 is depicted. There are two kinds of nodes representing respectively the base relations and the derived page schemes. Edges represent the dependency of a page scheme from a base relation, i.e. the fact that the relation is involved in the generation of the page scheme instances. If an attribute of the base relation is involved in the oid generation, its name is quoted in the edge label. Moreover, for each base relation, the current $\Delta$ are explicitly stored. Finally, each view node has a pointer to both the $\Delta$ of each of its base tables: it points the entry from which the next maintenance operation has to start, i.e. the first of the tuples corresponding to updates that have not been yet propagated to that page scheme instances. Of course, when maintenance of a page scheme is immediate, there is no need to manage those pointers for that view.

In Fig. 5.1 for example, the ClinicianPage view is connected with its base relations (PHYSICIANS, PATIENTS and EXAMS) and the connection with the PHYSICIANS relation is labeled with the name of the url attribute (PhysName). The view has also pointers to the $\Delta$ of each relation. In particular, each pointer refers to the first $\Delta$ entry that has to be used for the next view maintenance. Here, a “snapshot” is depicted where the dr. Mydirs tuple has been deleted from the PHYSICIANS table; and the tuples [Dumbys, F, 1-6-1998, Marquis] and [Lybrand, F, 1-3-1998, Marquis] have been inserted into the PATIENTS table, as a consequence of the modification of the old tuples. The latter are still waiting to be propagated to the relevant hypertext views.

The portion of a $\Delta$ starting from the tuple pointed by a view will be addressed as the Waiting Portion of the $\Delta$ for that view. After each maintenance operation, the pointers are moved to the end of the $\Delta$. An overall $\Delta$ update operation is needed of course, to delete from $\Delta$ all the tuples that are not anymore concerned by view maintenance operations.

The structure is light enough to be maintained without excessive overhead for the system because $\Delta$ entries are independent of the number of views. The pointer

---

4For the sake of simplicity, in Fig. 5.1 not all the $\Delta$ are depicted.
mechanism allows in fact to “share” each \( \Delta \) among all the views over the same relation. Moreover, at run-time, the number of updates is usually small with respect to relations.

Note that it is possible to adopt a different maintenance approach for each view. One can decide for example to regenerate the PatientPage instances after each transaction on the base tables, to keep early aware everyone in the hospital of changes in the health status of each patient; while adopting a batch approach, for example doing maintenance at midnight, for the instances of the DiagnosticianPage page scheme, because these pages are only accessed in the morning by the administrative staff of the hospital.

### 5.3.3 An Algorithm for the Maintenance of Page Scheme Instances

The algorithm design has then been based on page regeneration as the single maintenance operation. Each maintenance transaction corresponds essentially to the regeneration and/or removal of the instances of a page scheme. That is, given a page scheme, the algorithm produces all the needed manipulation statements to update the page scheme instances with respect to the tuples contained in the \( \Delta^+ \) and \( \Delta^- \) of the relation.

The algorithm produces essentially two kinds of page update statements: REMOVE statements and GENERATE statements, whose semantics have been described in Sec. 5.2.1.

In Sec. 5.2.2, one of the main problems in maintaining the nested views in case of deletions has been pointed out: some deletions correspond to nested tuple (i.e. pages) updates, while some others correspond to actual nested tuple deletions, that is, page
removals. The capability of managing the $\Delta^-$ of base relations allows also to provide an efficient solution to this problem.

Now, let an URL-relation (with respect to a page scheme) be a base relation in which at least one of the attributes are involved in the generation of the URLs of the instances of the page scheme. For what has been illustrated in Sec. 5.2.2, each tuple in each $\Delta^-$ of an URL-relation corresponds to the removal of at least one page. The tuples deleted from each URL-relation ($UR_i$) appear in the $\Delta^-$ of the relation, which is joined with the other URL relations to give a new relation ($UR_{\text{\tiny \Delta^{-}}} i$). From this relation the system, using the names of the URL attributes coming from the dependency graph ($UA_j$) and their values in the waiting portion of the $\Delta^-$ of the relation ($UV_{j,k}$), is able to extract the remove conditions (in the sequel they will be addressed as $U\Delta^-C_i$):

$$UR_{\text{\tiny \Delta^{-}}} i.A_1 = UV_{1,1} \text{ AND } \ldots \text{ AND } UR_{\text{\tiny \Delta^{-}}} i.A_n = UV_{1,n}$$

$$\text{OR}$$

$$\ldots$$

$$\text{OR}$$

$$UR_{\text{\tiny \Delta^{-}}} i.A_1 = UV_{m,1} \text{ AND } \ldots \text{ AND } UR_{\text{\tiny \Delta^{-}}} i.A_n = UV_{m,n}$$

where $m$ is the number of tuples in the waiting portion of the $\Delta^-$ and $n$ is the number of URL attributes of the relation. The algorithm produces one such condition for each URL-relation and uses them for the production of the following update instructions:

```sql
REMOVE < PageScheme >
WHERE $U\Delta^-C_1$

\ldots

REMOVE < PageScheme >
WHERE $U\Delta^-C_p$
```

where $p$ is the number of URL relations.

After page deletions, the pointers from the page scheme to the $\Delta^-$ of the URL-relations must to be moved to the end of the $\Delta^-$ itself to keep track that maintenance has been performed on that page scheme.

The $\Delta^-$ of base relations that are not involved in URL creation, together with all the $\Delta^+$, contribute to determine the set of pages to be generated again, as it will be clarified in the sequel. For each of such $\Delta$, the proper regeneration conditions are produced and they contribute to form the WHERE condition for the GENERATE statements.

Let us start with inserted tuples. If the instances of a page scheme are generated starting from a collection of base relations ($R_1, \ldots, R_q$), then the tuples inserted into a table (say $R_i$) appear in the $\Delta^+$ of the table, which is joined with the other base tables to give a new relation ($R_{\text{\text{\tiny \Delta^{-}}} i}$). From this relation the system, using the names of the URL attributes coming from the dependency graph, is able to extract the

---

5 A page scheme may in fact have in general more than one URL relation.
condition \((\Delta^+ C_i)\) to use for incremental regeneration of the page scheme instances with respect to \(R_i\):

\[
R_i \cup A \cup C_i = UV_{i,1} \quad \text{AND} \quad \ldots \quad \text{AND} \quad R_i \cup A \cup C_i = UV_{i,n}
\]

OR

\[
\ldots
\]

OR

\[
R_i \cup A \cup C_i = UV_{m,1} \quad \text{AND} \quad \ldots \quad \text{AND} \quad R_i \cup A \cup C_i = UV_{m,n}
\]

The generated statements to maintain the view with respect to inserted tuples are then the following:

\[
\text{GENERATE} \quad < \text{PageScheme} > \quad \text{WHERE} \quad U \Delta^+ C_1;
\]

\[
\ldots
\]

\[
\text{GENERATE} \quad < \text{PageScheme} > \quad \text{WHERE} \quad U \Delta^+ C_q;
\]

They generate the page scheme instances, using the conditions to regenerate only the pages corresponding to inserted tuples, as described by Expr. 5.1. The construction of the condition and the generation of update statements is repeated for each \(U \Delta^+\) of the base relations.

The case for deletions not involving URL attributes is similar but with a slight subtlety. As it has been illustrated, deletions on URL base tables correspond to physical deletions of hypertext files. Deletions on base tables whose attributes are not involved in URL generation, must correspond to the regeneration of the pages that contain the deleted values, which have to disappear from those pages. Then, \(\Delta^-\) of URL relations must not be taken into account for condition generation, because its entries have already been propagated as page removals: tuples deleted from the URL relations correspond to page removals as previously illustrated. The system will generate only \((q - p)\) conditions \((NU \Delta^+ C_i)\) corresponding to the non-URL base relations of the page scheme and, consequently, only \((q - p)\) GENERATE statements.

There is anyway some redundancy that may arise in such a maintenance process.

**Modifications.** Value modifications are handled as deletions followed by insertions. Hence, if a table tuple is updated, an entry is inserted in its \(\Delta^-\) and one in its \(\Delta^+\). Two different cases must be distinguished.

1. **The table is involved in URL generation.** A condition is produced for the corresponding REMOVE statement, and a different condition is produced for the GENERATE statement. The page identified by the old URL is correctly removed and then a new page is regenerated, which is identified by a new URL.

2. **The table is not involved in URL generation.** The same condition is produced from the \(\Delta^-\) and \(\Delta^+\) entry and two identical GENERATE statements are produced.
5.3. AUTOMATIC AND INCREMENTAL PAGE MAINTENANCE

Different updates generating the same condition. If two tuples inserted into (or deleted from) different tables originate the same condition, again, two identical Generate (or Remove) statements are produced.

These drawbacks can be avoided by the following optimization:

1. removal and generation conditions are produced all at once;
2. duplicates are eliminated from removal and generation conditions before producing the Generate and Remove statements.

The maintenance algorithm is based on the previously illustrated principles. Its definition and description follow.

**INPUT** The Page Scheme Dependency Graph of a derived hypertext, the underlying database and the WALG definition of the page scheme whose instances are to be maintained.

**OUTPUT** The maintenance statements for the page scheme instances.

**Sketch of the algorithm**

1. build conditions deriving from $\Delta^-$ of URL tables ($U \Delta^- C_i$);
2. eliminate duplicates from the above conditions;
3. produce

   ```sql
   REMOVE < PageScheme >
   WHERE $U \Delta^- C_1$
   ...
   REMOVE < PageScheme >
   WHERE $U \Delta^- C_t$
   ```

   where $t$ is the number of remaining conditions after duplicate elimination;
4. move to the bottom the pointers from the view node to the $\Delta^-$ of the URL tables;
5. build conditions deriving from $\Delta^-$ of non-URL base tables ($NU \Delta^- C_i$);
6. build conditions deriving from $\Delta^+$ of all the base tables ($\Delta^+ C_i$);
7. eliminate duplicates from the built conditions;
8. produce

   ```sql
   GENERATE < PageScheme >
   WHERE $NU \Delta^- C_1$
   ...
   GENERATE < PageScheme >
   ```
\[ \text{WHERE} \quad NU \Delta^- C_r \]
\[ \text{GENERATE} \quad \langle \text{PageScheme} \rangle \]
\[ \text{WHERE} \quad \Delta^+ C_1 \]
\[ \ldots \]
\[ \text{GENERATE} \quad \langle \text{PageScheme} \rangle \]
\[ \text{WHERE} \quad \Delta^+ C_s ; \]

where \( r \) and \( s \) are the number of remaining conditions after duplicate elimination.

9. move to the bottom the pointers from the view node to each \( \Delta^+ \) and \( \Delta^- \) of non–URL tables;

10. delete from each \( \Delta \) all the tuples that are in no more Waiting Portion.

Conditions are produced in the way illustrated in Sec. 5.3.3.

The algorithm meets the following requirements:

- each database update is propagated to the hypertext, because insertions and deletions are logged, and conditions are produced from logs;

- hypertext maintenance is incremental, because (i) each database update is propagated to the dependent views only, (ii) the use of conditions allows to restrict page (re)generation to the relevant pages only and (iii) the set of (re)generated pages is minimal, because conditions redundancy is avoided;

- finally, deletions corresponding to page removals are distinguished from those corresponding to page generations, thanks to information held by the Page Scheme Dependency Graph.
Chapter 6

Overview of Related Works
The concept of metadata is one of the most important among those who conduct Web–related research. The semistructured nature of the Web in fact, asks for methods and tools to enable resource discovery, content based search and retrieval, efficient indexing and information integration in a universe where different data formats, media, logical models and schemes, semantics and languages coexist. Many researchers have proposed approaches based on general description frameworks to challenge the above problems. The term metadata, in its broad meaning of ”data about data”, serves very often as an alias for different concepts, like for example catalogue, dictionary, keyword set, ontology, thesaurus, etc. The amount of work done so far in the field is then enormous. We will hence restrict our attention to those authors who have dealt with the use of metadata in health information systems, metadata for system integration and metadata in Web databases. We will also pay a lot of attention to a number of works in the field of view maintenance, which is very important in our framework, as has been illustrated in Chapp. 4 and 5.

6.1 Metadata in Health Information Systems

The use and management of metadata has been extensively explored by Medical Computer Science researchers. In particular, ontology and metadata systems have been developed for tackling integration, user–interfacing and retrieval problems.

Integration problems are faced for example by the authors of [52]. The project described here is aimed at managing the heterogeneities that arise in interoperating medical ISs. The approach is to define a domain ontology through the use of a reference model which allows for system evolution. Hyperlinked multimedia documents are used as an exchange mechanism. The authors show how the joint use of a data model for hypertext and a common metadata reference model may effectively contribute to solve the problem. Unfortunately, the management of metadata concerning the different levels of heterogeneity is not based on the same logical model and this results in an excessive specialization of the resulting federating system.

Chu and others [45] face as well, from an application point of view, the problem of semantic integration of health information. Their work is a valid example of how an Object Oriented approach to the management of metadata may be the right way to the solution. The approach we took in the MRBAQUE project extends this vision to the use of internet related technologies for tackling the problem in a distributed and heterogeneous architecture.

The members of the WIRM project [73] have developed a perl toolkit for managing and deploying multimedia data on the Web and have tested the approach by implementing an application for managing data about the human brain. An Object–Relational data model has been defined to allow the base data to be viewed as collections of objects. Unfortunately, the same model is not used to describe the application Web site, which is built using an API toolkit, without referring to any design model.

6.2 Metadata for System Integration

The process of integrating information systems consists essentially in resolving all the possible conflicts that may arise among the systems under integration. According
to the most common classification, these conflicts may pertain to the format, the schematic and the semantic level. Many approaches to the problem have then been based on the definition of metadata to describe the features of each system at each level. These metadata systems may or may not be managed by the same models and languages of the described data, but they are anyway used by specific system modules to convert data formats and resolve schematic and semantic conflicts.

Shet and Larson [97] for example present a reference architecture for federating distributed, heterogeneous and autonomous databases from both system and scheme viewpoint. One of the proposed architectural components is what they call Export Schema. It represents a subset of the scheme of one of the available databases, and it is expressed in a common data model. In our framework, the ADM scheme of a derived site may be viewed as an Export Schema of the underlying database and PENELlope as a filtering processor capable of materializing the instances of the schema and a schema representation at the same time. These materialized instances may then be easily accessed by another processor in charge of integrating multiple export schemas to produce the various external schemas for each class of federation users.

The process of defining an export view of a database according to a different scheme may be seen as part of the so-called mediation service. Mediators have been proposed for the first time by Wiederhold [110]: they are software modules that mediates between the workstation applications and the databases. Many other subtasks are involved in the mediation process and all of them make use of metadata describing data semantics, interpretations, aggregations and other classes of useful information. These first guidelines for a mediator architecture have been extensively used for the development of mediator–centred integration frameworks and systems [44, 5, 84, 91, 104]. The concept of mediator has reached such an importance that even a proposal for standard mediator languages has been set by Buneman and others [42]. The ARANeus project [1] has been one of the first projects aimed at studying the problem of mediation in the Web context. We believe that the models and tools we proposed represent a complete enough suite of basic instruments to be used for the construction of sophisticated Web information brokers.

6.3 Metadata for Web Databases

The studies on metadata in Web databases have been so far devoted to the definition of models and tools for describing and accessing information about the structure and behaviour of Web data. The common goal is to develop services that take full advantage of the connectivity of the Web.

6.3.1 A Classification

Keith Jeffery has proposed an interesting and complete classification of metadata for Web databases [74]. He distinguishes metadata in schema, navigational and associative. Schema metadata are an intensional description of extensional instances. Navigational metadata provide information on how to get to an information resource. While associative metadata provides additional information for application assistance. The solutions proposed in this thesis deal with schema metadata. They allow for
validation through schema and constraints, performance optimization, answer consolidation and integration across heterogeneous data sources.

6.3.2 Standards

There have been many attempts to standardize metadata structure and content for specific application areas.

The increasing requirement for interworking among systems handling grey electronic literature has caused the internet community to propose the Dublin Core [4]. It is a set of metadata elements that is proposed as the minimum number of elements required to facilitate the discovery of document–like objects in a networked environment such as the Internet. While the Dublin Core falls into the class of navigational metadata, we believe that a Web authoring system may take advantage from our tools for producing Dublin Core–compliant documents. In fact, provided that all the necessary information is stored in the database, it is sufficient to add to the generated site a page scheme whose attributes are the elements of the Dublin Core and map the database data onto the HTML pages.

We have extensively talked about XML in Chap. 4. Starting from its definition, a number of tagging systems for specific application areas have been proposed [19]. As we have shown, using Penelope it is possible to map database data into generic XML documents, hence allowing for the publication of datasets coming from those specific areas.

Also using XML as the base syntax, a number of specific metadata systems are being developed at the W3C [7]. The Resource Description Framework (RDF) for example is a model for representing named properties and their values. These properties serve both to represent attributes of resources and to represent relationships between resources. We believe that the principles we described about the correspondences between ADM and XML, which will serve as a basis for some of the future developments of the Araneus project, are applicable to RDF as well.

6.3.3 Metadata for Web Site Design

The importance of metadata has been pointed out also by some of the authors who study design methodologies for Web sites.

For example, in the framework of the STRUDEL project [56] metadata about the various information sources that are involved as the base data sources of a new site are maintained and queried using the same graph model and the same query language of the actual data. This is somehow similar to the solution proposed in this thesis, but we believe that the use of a less structured model, as a graph–based one, for modeling the structure of the site, may lead to some difficulties in the page maintenance activity, when and if materialized pages are generated.

The developers of the ConceptBase system [106] stress the value of metadata during the lifecycle of a Web–based Information System, especially if they are presented at three levels of granularity (global, local and atomic) and orthogonally organized in three levels of abstraction (conceptual, logical and physical). In the framework of the Araneus project a design methodology for Web sites has been proposed [32]. It is based on the same three–levels approach (conceptual, logical, physical) to site design,
but it doesn’t cover the global aspect of a WBIS, that is the synthesis of all the local information spaces of the component sites.

### 6.4 Materialized Hypertext Maintenance

The approach to metadata publishing presented in this thesis is based on the concept of materialized derived hypertext. They are Web pages containing data coming from an underlying database and they are either stored on the Web server or pushed on the client machine. The materialized approach requires of course some maintenance activities. In this section we will give an overview of some interesting work related to hypertext view generation and maintenance.

#### 6.4.1 Hypertext View Generation.

The motivation of the work presented in this thesis is the development of the Araneus Web Base Management System [1].

A number of systems have been proposed to deal with the contemporary presence of structured and semistructured data. Some of them specifically address the problem of generating a Web site as a view over a set of data sources [55, 53, 58, 92, 101]. None of them support incremental maintenance and their view generation mechanism lacks the notion of site scheme. They are either based on a graph-based model [55, 92, 101], or on a model derived from hypermedia authoring [53, 58].

There are also some commercial database systems (see, for example, [6, 10]) providing Web page generation facilities. They are however based mainly on pull techniques and they also lack of the notion of site scheme.

#### 6.4.2 Incremental Maintenance of Materialized Views.

The principle of incremental maintenance has been previously explored by several authors in the context of materialized database views [38, 37, 43, 68, 81, 69, 77, 78, 107].

Gupta and Mumick [69], provide an useful overview of problems, techniques and implementations of materialized views. Unfortunately, the specific issues connected with nested and oid-based views are not discussed.

The authors of [38] were the first ones to introduce the concept of materialized view on relational databases. They propose an incremental maintenance algorithm for SPJ views\(^1\) and they introduce a method to automatically find out if a given view is independent of a database update. Those principles are extended and carried through in [37], where the emphasis is on the view independence problem. Though a similar approach could be used in the context of hypertext views, for detecting which page scheme is affected by a database update, the explicit representation of dependencies between each page scheme and its base tables has been preferred to avoid SPJ view limitation and for efficiency reasons.

Ceri and Widom [43] propose a facility to automatically derive production rules to maintain a materialized view. The idea of having a manipulation language to

---

\(^1\)In SPJ views the only allowed relational operators are Selection Projection and Join.
propagate updates to views is similar to the one presented in this thesis, which extends that idea to the maintenance of hypertext views.

The authors of [68] propose the Counting Algorithm for incremental materialized view maintenance. Though their major aim was to provide a framework to deal with the management of view duplicates, their proposal has been considered as a starting point by successive works aimed at maintaining materialized views in nested data models.

Kawaguchi and others [77, 78] discuss materialized views defined over a nested data model. They use a Datalog-like view definition language and they extend the Counting Algorithm of [68] to handle views over nested base tables. The present proposal deals with issues arising in the different context of nested oid-based views that are stored as hypertext pages (i.e. text files). In particular: each page (view tuple) is a nested object and a different maintenance approach is considered, which (re)generates the whole set of update-interested pages.

The orthogonal problem of deferring maintenance operations, thus allowing the definition of different policies, has been studied by the authors of [47, 76] and [48]. One of the future directions of the ARANEUS project will explore such issues in the framework of hypertext views.
Chapter 7

Conclusions and Open Problems
This thesis presented an approach to metadata publishing and maintenance in the Web context. We assume that the site pages contain data coming from an underlying database and their logical structure is described according to a specific model: ADM, the ARANEUS Data Model, which allows for the definition of what we call data-intensive Web sites [32]. The pages of the site may be mapped on the database and automatically generated using PENEOLE, a declarative language composed by a definition language (PDL, the PENEOLE Definition Language) and a manipulation language (PML, the PENEOLE Manipulation Language), in the spirit of DBMS’s.

A system prototype, the ARANEUS Web Base Management System (WBMS) has been developed to prove the satisfiability of the proposed solutions. The kind of metadata we deal with are information about the logical structure and the attribute names of the pages. The current version of the system allows in fact for the embedding of such metadata in the comments of a HTML page.

In order to allow external applications for accessing these metadata a materialized approach has to be adopted for page generation: the produced HTML files are actually stored in the same server where the HTTP server is running.

These solutions have partly been inspired by the experience we had in the framework of the MRBRAQUE project, aimed at developing a system for the management of hypermedia medical reports.

We here discuss some relevant issues and open problems.

7.1 Development of Web-based Medical Applications

The MRBRAQUE project showed how the Web technologies may be very helpful in making quicker and easier the development of semantically rich medical applications. While on one hand this experience helped us in devising a new, more general research direction and starting the ARANEUS project, on the other hand the MRBRAQUE project had a further evolution and posed some problems specifically for the medical computer science context.

The introduction of the information technology for the realization of health information systems has been in fact so far characterized by etherogeneous approaches, mostly aimed at satisfying local and specific requirements. In such a context the definition of common reference models for information interchange is needed, together with common frameworks for data representation and interpretation. In fact, if in the computer networking field such standards have already been defined (like ISO/OSI, TCP/IP, etc.) and widely used, more research is necessary for achieving a correct terminological and structural interpretation of medical information.

The aim of a medical standard is to allow for an as wide as possible access by the health operators to an ever greater amount of data from different sistems and formats, overtaking the limits of local and specific solutions. In particular, the use of a standard as a meta information model may be fundamental not only for the realization of tools for accessing distributed and heterogeneous information, but also for the integration of autonomous legacy systems.

In the field of the management of biomedical images there are a number of ongoing standardization projects. Among them, DICOM (Digital Imaging and Communication...
in Medicine) [25] is being internationally accepted, also because it is the result of a cooperation effort between industrial and professional associations.

In this context, we started a task aimed at making the MRBrAQue system \textit{DICOM-compatible}. In particular, we abstracted a conceptual model over the standard and defined a mapping between the two models. This allowed us for the development of mapping functions to transmit and receive hypermedia reports in DICOM format.

## 7.2 Publishing of Web Metadata

One of the topics that attracts the interest of many researchers and practitioners of the Web and databases fields is at the moment XML. Most of the efforts are aimed at modeling XML repositories and defining query languages for querying and transforming XML sources [21].

One of the current research directions of the Araneus project is to explore XML, both as a syntax for metadata publishing and as a document model to be queried and restructured. The authors of [86] show that XML modeling primitives may be considered as a subset of ODMG enriched with union types and XML repositories may in principle be queried using an OQL–like language. There are however two major peculiarities of XML data that force to re–consider some features of the query process:

1. **XML repositories are remote and autonomous** XML repositories will probably be mostly available on the Web, rather than locally stored; therefore, they will be autonomous, i.e. not under the control of the query system; moreover, the repository manager will update the repository, or even DTDs, without notifying remote users.

2. **XLinks are untyped** As has been pointed out in Chap. 4, XLink does not allow typed links; although it is somehow probable that the final XLink standard will include typed links, still it will be possible to use untyped links in DTDs.

These considerations lead to some interesting research issues:

- **Schema Derivation** Query processes based on languages like Lorel [24], WebOQL [28] or Ulixes [29] if applied to XML, would rely on a database abstraction, i.e. a scheme, over the XML repository. If we do not want an a–posteriori derivation of such a scheme when untyped links are used, some abstract description of the structure of the whole repository should be defined. One possible solution may be to define a specific XML syntax for the definition of the structure of an XML repository. In the spirit of relational catalogues, in which a description of the database structure is given using the same logical model of the database. Such an effort is being made in the framework of XMLData [22], a W3C proposal for the abstract definition of a data source structure and constraints.

- **Dealing with Updates in the Schema** Even the structure of the repository may change and an external query application will not have any notice of that
in advance. That means that the above metadata about the structure of the repository should be efficiently updated as well.

- **Dealing with Updates in the Data** In order for the above query systems to answer queries there are two possible alternatives: (i) the system may populate classes by loading objects into a local database, materializing therefore the original data into the local database; (ii) or it may leave documents at their original position, and download only the ones needed to answer the query at query-execution time. Each of the two approaches has of course some obvious advantages and disadvantages.

In particular, the former approach does not guarantee that query answers are up-to-date. Also, the database needs to be periodically maintained in order to reflect changes at the repository. Therefore, maintaining the database requires in principle to download the whole repository, which in some cases may have unacceptable costs.

There is a strong need for incremental view maintenance techniques for XML repositories, similar in principle to those presented in this thesis, but where base data are an XML repository which should maintain suitable meta-information to be accessed by external query systems in order to easily detect changes and update the local database.

### 7.3 Maintenance of Materialized Web Sites

In previous chapters we showed how one of the most important problems in managing information repositories where structured and semistructured data live together is to provide applications with the ability of managing the materialization of such repositories.

Our approach to the definition and generation of a database-derived Web site resulted particularly suitable for defining a framework for incremental updates of the site. In fact, the capability of formally defining mappings between base data and derived hypertext has allowed to easily define hypertext updates in the same formalism.

In order to provide the system with the ability of propagating database updates to the relevant pages, a Data Manipulation Language has been defined, which allows to automatically generate or remove a derived site or a specific subset of its pages. The presence of a DML for pages allowed the definition of an algorithm for their automatic maintenance. The algorithm makes use of a specific data structure that keeps track of view-table dependencies and that logs table updates. Its output is a set of DML statements to be executed in order to update the hypertext.

The algorithm may be used for the implementation of different view maintenance policies. It allows to defer maintenance to a later time and to perform it in batch mode by using the table update logs and implementing proper triggering mechanisms.

There are a number of connected problems that may deserve further investigations.

- Defining the most suitable maintenance policy for each page class involves a site analysis process to extract data about page accesses and, more generally, data about site quality and impact.
• If a deferred policy is chosen for a given class of pages, this causes part of the hypertext to become temporarily inconsistent with its definition. Consequently, transactions that read multiple page schemes may not execute in an efficient manner. The concurrency control problem needs to be extended to the context of hypertext views and suitable algorithms have to be developed. In particular, the problem of avoiding *dangling links*, i.e. links pointing nowhere because the formerly target page has been removed, should carefully be addressed.

• Performance analysis are needed in order to show how, in both the above approaches, transaction overhead and view refresh time are affected. The results of these analysis should be used to define a set of *system-tuning* parameters to be used by site administrators for optimization purposes.
7 Conclusions and Open Problems
Bibliography


[84] Madnick, S. From vldb to vmldb (very many large data bases): Dealing with large-scale semantic heterogeneity. In International Conf. on Very Large Data Bases (VLDB’95), Zurich (1995), pp. 11–16.


