A Four Level Architecture for Hypermedia Database Management Systems

DISSERTATION

Zur Erlangung des akademischen Grades eines
Doktors der Sozial- und Wirtschaftswissenschaften

Eingereicht an der
Johannes Kepler Universität Linz
Institut für Wirtschaftsinformatik
Data & Knowledge Engineering

Betreuung:
o. Univ.-Prof. Dipl.-Ing. Dr. Michael Schrefl
o. Univ.-Prof. Dipl.-Ing. Mag. Dr. Gerti Kappel

Linz, März 1998

Verfaßt von:
Thomas Prückler
Landwiedstraße 11
A-4020 Linz

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, daß ich die vorliegende Dissertation selbständig und ohne fremde Hilfe verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht verwendet und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen deutlich als solche gekennzeichnet habe. Diese Dissertation habe ich bisher weder im Inland noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt.

Linz, am 24. 3. 1998 ....

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die zum Gelingen dieser Arbeit beigetragen haben.

Ein ganz besonderer Dank gilt Prof. Michael Schrefl, der durch zahlreiche Diskussionen mit vielen wertvollen Anregungen einen wesentlichen Beitrag zu dieser Arbeit geleistet hat und der mir immer als "Prüfstein" meiner Ideen zur Verfügung stand. Bei Prof. Gerti Kappel bedanke ich mich für die Übernahme der Zweitbegutachtung.

Ich habe am Institut für Wirtschaftsinformatik der Universität Linz (Forschungsschwerpunkt Data & Knowledge Engineering) ein anregendes und angenehmes Arbeitsklima erlebt. Dafür bedanke ich mich bei allen Kolleginnen und Kollegen am Institut: bei Ingeborg, Peter, Werner, Peter, Roman und ganz besonders bei Margit. Mein spezieller Dank gilt meinem *lieben* Kollegen Günter, ohne den es viele anregende Momente nicht gegeben hätte.

Besonders danke ich meinen Eltern Brigitte und Johann, die mich über all die Jahre hinweg unterstützt haben.

Kurzfassung

Diese Dissertation beschreibt die **Hyper**media **D**atabase Management System (HyperD) Architecture für hypermediale Datenbanksysteme sowie die zur Implementierung notwendigen Klassen. Mit der HyperD Architektur kann sowohl die Struktur als auch der Inhalt von hypermedialen Daten auf konsistente, flexible, und objektorientierte Weise modelliert werden. Die HyperD Architektur kann auf einfache Weise erweitert werden, einerseits um neue Medientypen speichern zu können und andererseits um anwendungsspezifische Informationen modellieren zu können. Weiters beschreibt diese Dissertation Kernpunkte der Implementierung eines Prototypes für Bilder.

Ein Schwerpunkt der Arbeit besteht darin zu zeigen, daß physische und logische Datenunabhängigkeit sinnvoll in hypermediale Datenbanksysteme integriert werden kann. Der Ansatz zur physischen Datenunabhängigkeit berücksichtigt die Besonderheiten multimedialer Daten und erlaubt eine kontrollierte Wiederverwendung multimedialer Objekte. Der Ansatz zur logischen Datenunabhängigkeit ist sehr flexibel und erlaubt eine klare Trennung von anwendungsspezifischem Wissen und generellem Wissen.

Weiters beschreibt diese Dissertation, wie die HyperD Architektur um das Konzept des Corresponding Information Content (CIC) erweitert werden kann. CIC erlaubt die Modellierung von Menge und Qualität der in einzelnen Medienobjekten enthaltenen Informationen in Bezug auf die Informationen in anderen Medienobjekten.

Abstract

In this thesis the **Hyper**media **D**atabase Management System (HyperD) Architecture for hypermedia database management systems and the classes needed to implement the architecture are described. The HyperD architecture allows to model the structure as well as the content of hypermedia data in a consistent, highly flexible, and object oriented manner. The HyperD architecture can be extended easily to accommodate new media types and is flexible enough to model domain specific information. This thesis also describes implementation issues regarding a prototypical implementation for images.

One focus of the work is to show that the concept of physical and logical data independence can be applied beneficially to hypermedia database management systems. The approach to physical data independence presented here takes the characteristics of multimedia data into account and allows a controlled reuse of multimedia objects. The approach to logical data independence presented in this thesis is highly flexible and allows a clean separation of application specific and general knowledge.

This thesis also shows how the HyperD architecture can be extended with the concept of Corresponding Information Content (CIC). CIC allows to model the amount and quality of the pieces of information in one media object in relationship to the pieces of information in another media object.

Contents

1 Introduction **1**

1.1 Motivation . . . 2

1.2 Scope . . . 4

1.3 Architecture Overview . . . 8

1.3.1 Storage Level . . . 9

1.3.2 Media Level . . . 9

1.3.3 Content Level . . . 10

1.3.4 Presentation Level . . . 10

1.4 Structure of this Thesis . . . 11

2 Introductory Remarks **13**

2.1 Definitions . . . 13

2.1.1 Hypertext . . . 14

2.1.2 Multimedia . . . 17

2.1.3 Hypermedia . . . 20

2.1.4 Technical Terms . . . 23

2.1.4.1 Node . . . 24

2.1.4.2 Link . . . 26

2.1.4.3 Further Hypermedia Definitions . . . 29

2.1.5 General definitions . . . 30

2.2 Notation . . . 32

2.2.1 Classes . . . 32

2.2.2 Objects . . . 33

2.2.3 Relationships . . . 34

2.2.4 Aggregation and Inheritance Relationships . . . 37

3 Background and Related Work **41**

3.1 Related Areas 41

3.1.1 Media Server 42

3.1.1.1 Research Issues for Continuous Media Recording and Retrieval 43

3.1.1.2 Selected Research Results for Continuous Media Recording and Retrieval 44

3.1.1.3 Research Issues for Managing Digital Multimedia Storage 46

3.1.1.4 Selected Research Results for Managing Digital Multimedia Storage 47

3.1.1.5 Discussion 49

3.1.2 Information Retrieval 49

3.1.2.1 Research Issues for Information Retrieval 50

3.1.2.2 Selected Research Results for Information Retrieval 52

3.1.2.3 Discussion 56

3.1.3 Document Data Models/Interchange Standards 56

3.1.3.1 SGML/HyTime 56

3.1.3.2 MHEG 58

3.1.3.3 ODA 59

3.1.3.4 Discussion 60

3.2 Hypermedia Databases and Data Models 61

3.2.1 Semantic and Object Oriented Data Modeling 61

3.2.2 Survey Articles 62

3.2.3 Hypermedia Data Model and DBMS 63

3.2.3.1 Dexter Hypertext Reference Model 63

3.2.3.2 Hypertext Abstract Machine 65

3.2.3.3 Neptune 66

3.2.3.4 Garg's Set Theoretic Model 66

3.2.3.5 Intermedia 67

3.2.3.6 Hyperbase 68

3.2.3.7 Hyperform 69

3.2.3.8 MODEL 71

3.2.3.9 ABC distributed hypermedia collaboration 72

3.2.3.10 HyperBase3 73
3.2.3.11 HyperStorM 75
3.2.3.12 GRNL model 75
3.2.3.13 Enhanced Object-Relationship Model 78
3.2.3.14 Trellis Model 79
3.2.3.15 Transient Hypergraph Model 81
3.2.3.16 Hypergraph Data Model 82
3.2.3.17 Hypertext Design Model and Successors 83
3.2.3.18 Hypermedia Data Model 89
3.3 Commercial Database Management Systems 92
3.3.1 Oracle 93
3.3.2 ObjectStore 94

4 Data Independence 97
4.1 Data Independence in Conventional DBMS 98
4.1.1 ANSI/X3/SPARC DBMS Architecture 98
4.1.2 Logical Data Independence 101
4.1.3 Physical Data Independence 101
4.2 Data Independence in Hypermedia DBMS 101

5 HyperD Architecture 109
5.1 Clarifications and Remarks 111
5.2 Storage Level 112
5.2.1 Chunk 113
5.2.2 InternalMedium 113
5.2.3 Relationship between Internal Media and Logical Media . 115
5.3 Media Level 118
5.3.1 Logical Medium 118
5.3.2 Design Considerations 120
5.3.3 Combined Logical Medium 122
5.3.4 Benefits 122
5.4 Content Level 123
5.4.1 Information Content Objects and Items of Information . . 123
5.4.1.1 Differences between IOIs and ICs 128
5.4.1.2 Extent of IOIs 128

5.4.2 Relationships in the Content Level 129
5.4.3 Whole-Part Relationships 131
5.4.3.1 Composition: IOI-Composite 132
5.4.3.2 Decomposition: IOI-Extract 138
5.4.3.3 Characteristics of Extract- and Composite-Relationships 139
5.4.3.4 Relationship Propagation 142
5.4.3.5 Whole-Part Relationships in existing Hypermedia Data Models and Hypermedia DBMS 145
5.5 Presentation Level 146
5.5.1 Purpose of the Presentation Level 147
5.5.2 Object Model of the Presentation Schema 149
5.5.2.1 Example 151
5.5.2.2 Presentation Items 151
5.5.2.3 Link Definitions and Presentation Links 153
5.5.3 Meta Schema 158
5.5.4 Application Specific Knowledge 159
5.5.4.1 Guided Tours 160
5.5.4.2 Table of Contents/Indices 161

6 Data Independence in HyperD 163

6.1 Storage Independence 164
6.2 Media Independence 166
6.3 Content Independence 167

7 Corresponding Information Content 171

7.1 Basic concept 172
7.1.1 Nature of Hypermedia Data 172
7.1.2 Redundancy in Hypermedia Data 173
7.1.3 Corresponding Information Content 174
7.2 CIC-Relationships 175
7.2.1 Content 176
7.2.2 Level of detail 179
7.2.3 Combination of CIC-indicators 180
7.2.4 CIC-tags of Relationships 180

7.2.5 Inference of CIC-tags 180
7.3 CIC relationships in the HyperD Architecture 183

8 Prototype 187

8.1 Illustra 187
8.1.1 Illustra Database Management System Server 188
8.1.2 Illustra Image DataBlade 188
8.1.2.1 Image Formats 189
8.1.2.2 Classification of Operators 190
8.1.2.3 Operators in the Image DataBlade 190
8.1.2.4 Performance Measurements 191
8.2 Implementation 202
8.2.1 Implementation Decisions 202
8.2.2 Insertion of New Media Objects 203
8.2.3 Creation and Deletion of Secondary Internal Media 204
8.2.4 Retrieval of Media Objects 205
8.2.5 Commutativity of Operators 206
8.2.6 Execution of Queries 207

9 Conclusion 213

List of Figures 216

List of Tables 218

Bibliography 219

Curriculum Vitae 244

Chapter 1

Introduction

As a large amount of hypermedia data is collected the question of storing this kind of data in hypermedia database management systems arises. One of the problems encountered is how to separate applications from the data so that the same data can be used by many applications and that changes to data organization have—besides execution-time—no effect on applications. In conventional databases this problem is solved by the concept of data independence. In this thesis we argue that the concept of physical and logical data independence as it is used in conventional database management systems can be applied beneficially to hypermedia database management systems as well.

So far, while the demand for physical data independence as well as logical data independence has been voiced in a number of papers, only partial solutions have been proposed. We give an overview of the concepts associated with logical and physical data independence and examine how these concepts are supported by existing hypermedia systems. We show which aspects of data independence can be distinguished in hypermedia database management systems and we describe an architecture of a hypermedia database management system which deals with physical and logical data independence.

We describe a powerful way to incorporate multimedia data in a hypermedia network of links and nodes. This approach supports links and nodes as subclasses of user defined classes with additional meaning and also simple nodes and links.

All these issues are being dealt with in the **Hyper**media **D**atabase Management System (HyperD) Architecture, which represents the core of this thesis. The work presented in this thesis is the consolidation and extension of the work presented in [Prückler and Schrefl, 1995; Prückler and Schrefl, 1996; Prückler and Schrefl, 1997b; Prückler and Schrefl, 1997a].

This work describes an architecture, not a full implementation. This architecture is intended to be a core architecture, which can be extended easily using the object oriented approach to handle new kinds of multimedia data and to model data in a domain specific manner. We show how an existing database management system can be extended to meet this architecture. In particular, we will discuss implementation issues of a prototypical partial implementation, which extends the object-relational database management system Illustra with data independence for images. In a master thesis [Lisberger, 1998] an implementation of the architecture for images and text is described.

This chapter is structured as follows: Section 1.1 explains the motivation for the work presented in this thesis. Section 1.2 describes which goals were taken into consideration when the HyperD architecture was defined. Section 1.3 gives a brief overview over the architecture presented in this work. Section 1.4 describes the structure of this thesis.

1.1 Motivation

The nature of multimedia and hypermedia data differs substantially from alphanumerical data [Campbell and Chung, 1995]:

> Multimedia data is used for its presentation and representation capabilities rather than as a computational data type. Multimedia data

> captures, abstracts, and represents humanly presentable information. ... However, these presentational characteristics inhibits its use for computational processing which extracts additional information from existing information. Future computational functions will start emerging, but multimedia data will not be processed like financial data.

Other differences to conventional data include the necessity to model temporal relationships and the emphasis on the navigational aspect. Even the kind of navigation can be different from the navigation as found in conventional or object oriented databases: in conventional or object oriented databases navigation, i.e. moving from one instance to another, is always a process initiated or at least perceived to be initiated by the user. In hypermedia databases, playing an object can also be navigating, because sequentially playing the parts of an object can cause moving to other objects and different links can be available as the playback progresses [Garzotto and Paolini, 1991]. For example, a video showing a walking tour through Linz can have links to different sights at different points in time. The emphasis on navigation causes the need for a powerful way to model semantic relationships.

While multimedia data can be stored in conventional database management systems as a conventional attribute, this solution fails to achieve the same quality of storage for hypermedia data as it is achieved for formatted, alphanumerical data. As will be shown later, especially logical and physical data independence is not achieved and the full meaning of hypermedia data cannot be modeled adequately.

For these reasons, conventional data modeling and conventional database management system architectures are not suitable for hypermedia data and thus a new approach is needed. But, as discussed in Chapter 3, existing hypermedia and hypertext data models also fail to capture the full meaning of hypermedia data in the context of a database management system. Those data models primarily model single applications, or the data of a single application, or a small group of such applications, but are not able to model multimedia data in an application independent way.

In addition to the issues discussed above, [Maurer *et al.*, 1993b] identifies the following problems of hypermedia systems:

1. An unstructured network of links causes unnecessary cognitive overhead for the user who gets "lost in hyperspace".
2. The meaning of a link attached to a node is not visible to the user.
3. Deletion of nodes can cause serious consistency problems with dangling links.

Many solutions to solve these problems have been proposed. For example, in [Botafogo *et al.*, 1992] methods for identifying hierarchies and the computation of useful metrics are described and in [Utting and Yankelovich, 1989] some tools like global and local maps, different kinds of overviews and time charts are evaluated. But none of these proposals addresses the problem in an integrated, comprehensive manner.

As discussed above, conventional database management systems as well as existing hypertext and hypermedia data models fail to comprehensively model hypermedia data in a database management system.

1.2 Scope

This section describes the scope of the work presented in this thesis, the goals which we want to achieve and the limitations of the approach presented here.

In [Ghafoor, 1995] the author argues, that because of the heterogeneous nature of multimedia data, systems which intend to store, transport, and manage such data must have more capabilities than conventional information management systems. Ghafoor identifies the following issues as specification requirements for multimedia data:

1. *Development of formal semantic modeling techniques for multimedia information, especially for video and image data.* Important requirements include rich capabilities for abstracting multimedia information, the ability to capture semantics expressed by multimedia data, consistent representations of the objects portrayed in multimedia data and the spatio-temporal relationship between these objects.

2. *Design of powerful indexing, searching, and organization methods for multimedia data.* Manually browsing through multimedia, playout management, and processing multimedia data for content-based retrieval has to be accelerated by appropriate primary and secondary storage structures.

3. *Development of models for specifying the media synchronization/integration requirements.* Different (mono-)media have to be synchronized efficiently and consistently. The information about synchronization has to be integrated with higher-level information abstractions at a schema level.

4. *Designing formal multimedia query languages.* Important design considerations include the ability to express arbitrarily complex semantic and spatio-temporal information and to relate this knowledge to the multimedia data itself.

5. *Development of efficient data-placement schemas for physical storage management.* These schemas allow efficient access to multimedia data for playout to the user.

6. *Design and development of suitable architecture and operating system support.* Operating systems have to be extended to be able to deal with the real-time requirements of multimedia retrieval and playback. Because multimedia information and platforms are heterogeneous, an architecture of a general multimedia database management system is needed which supports a rich set of data-management functionalities, a diverse set of processing techniques and algorithms, and a type system that can be extended easily with new types of multimedia data and associated processing techniques.

7. *Management of distributed multimedia databases.* A further requirement is the ability to cope with the special challenges of distributed systems, for

example location transparent access and distributed real-time delivery.

Our approach focusses on the first four of the issues identified above, with the exception that we do not cover issues related to searching of multimedia data (cf. item 2). While we do not define a formal multimedia query language (or, in this case, a hypermedia query language), such a language can easily be added on top of the architecture by reusing the constructs provided for the creation of views (see Chapter 5). We do not describe an index which allows to locate a certain piece of media data quickly—this is done by other constructs—but one which is fulfills the original goal of indexes, i.e. speeding up query processing. Technical aspects (cf. items 5-7), for example operating system support, data-placement schemas, or management of distributed multimedia databases, are not considered.

In addition to the requirements described above, we have identified the following requirements:

- Any data can be inserted into the database, i.e. the data should not have to be prepared in a special manner. The database management system should also be able to store objects with a preplanned structure which were created for a specific purpose.
- Bottom-Up as well as Top-Down modeling should be possible.
- The architecture should be implementable on top of an existing database management system.
- The database management system is designed for a *large* number of objects, used by a large number of applications. For this reason, data modeling is not application specific, but should model the content of the data as well as other application independent knowledge. Application specific aspects should be modeled separately in the top level of the database management system.
- The database management system does not extract knowledge from multimedia data, but it can be used to store the results of knowledge gathering

processes or manual entry of knowledge and to display these results. The inference of CIC-tags as described in Chapter 7 can be regarded as (rudimentary) knowledge manipulation.

Additionally, the following general goals are taken into account when defining the architecture:

- Data independence: Achieving data independence is one of the most important goals of the HyperD architecture. For a detailed discussion see Chapters 4 and 6.
- No redundancy: as explained in Chapter 7, for hypermedia data additional issues regarding redundancy have to be considered. As will be shown in Chapter 7, redundancies cannot always be avoided. The HyperD architecture provides constructs to minimize and control redundancy.
- Updateability: since sometimes redundancies cannot be avoided for hypermedia data, updating information becomes more difficult. This problem is addressed by providing data independence and, to a minor extent, by the constructs presented in Chapter 7.
- Reusability: since hypermedia data can be very expensive to produce, think for example about a professionally produced public relations video, hypermedia data has to be considered a valuable resource and should be used in a variety of ways. The HyperD architecture provides a framework for controlled reuse of hypermedia data.
- Understandability: Semantic modeling can be used to clarify the relationships between data objects.
- Extendibility: the architecture should allow extensions, especially the introduction of new media types.

We note the following restrictions for the HyperD architecture:

- Issues related to media server (especially video-on-demand and related areas), information retrieval, and document data models are outside of the scope of this work. Those issues and how they relate to the work presented here are discussed in Section 3.1.

- Versioning. Some attempts have been made to incorporate versioning features into hypermedia systems, see e.g. [Østerbye, 1992; Haake, 1992; Durand *et al.*, 1994]. Various versioning schemes can be incorporated into the HyperD architecture.

- Security. Until now, security issues for hypermedia data have not been dealt with thoroughly, with the exception of security on the level of nodes (see e.g. [Campbell and Goodman, 1988; Anderson *et al.*, 1994]). As some nodes are large, this level is to coarse for many applications. Also links have to be considered for security issues, because an user can deduce information from the existence or name of a link. Security can be incorporated easily in the HyperD architecture. The rich modeling capabilities allow security features which take the content of the media data into account, e.g. features which hide all information which is similar or related to a specified object.

- Issues related to user interface considerations. The architecture presented here was not meant to be used by an end-user to build views interactively, but for applications which access and change the database. For this reason considerations about user interfaces, support and tools for end-users are no primary concern for the work presented here.

1.3 Architecture Overview

This section gives a brief overview of the architecture described in this thesis. This overview is intended to provide a first impression of the architecture and to enable the reader to understand how the related work discussed in Chapter 3 relates to the work presented in this thesis. The architecture is described in detail in Chapter 5.

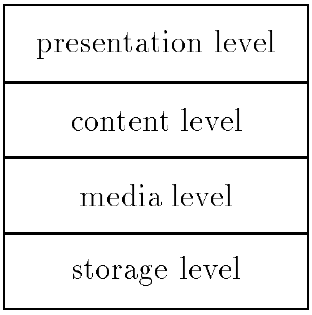

Figure 1.1: Highlevel Hypermedia DBMS Architecture

The HyperD architecture is shown in Figure 1.1. From bottom to top, the architecture consists of the storage level, the media level, the content level, and the presentation level. Each level has a schema describing the data structures used to model the information contained in the level.

1.3.1 Storage Level

The storage schema describes the actual storage structure of the multimedia data. Information stored in this level includes for example the number of chunks a multimedia object is composed of, its storage location, its storage format, its sampling rate, and the multimedia data itself. In addition to the data inserted into the database, secondary storage structures can be created to accelerate query processing.

1.3.2 Media Level

The media schema describes multimedia data disregarding the storage structure. Each piece of multimedia data inserted into the database is described by one object of this schema.

At this level only some basic properties of multimedia data are known, e.g. the length of a video recording, most other properties are hidden in the storage level. The functionality implemented at this level is responsible for query processing, query optimization, and for the management, e.g. instantiation and deletion of secondary storage structures, of the storage level.

1.3.3 Content Level

The content schema describes the objects contained in and described by the multimedia data, especially which media objects contain which information and how the information contained in one media object is related to the information contained in other media objects. The content level contains the information which is modeled (in part) by existing hypermedia data models (see Section 3.2 for a discussion of hypermedia data models). In the context of the HyperD architecture the content of a piece of multimedia data is the meaning conveyed to the user when she reads or listens to or looks at the data. Note: this usage of the term "content" is different from the usage of the term in the area of information retrieval and content-based retrieval (see Section 3.1.2).

The content level uses Information Content Objects (IC-Objects, introduced in [Prückler and Schrefl, 1995]) to model the information contained in multimedia data objects. IC-Objects represent abstract entities of information. Concrete information about an IC-Object is described by Items of Information (IOI). IOIs describe the information which is conveyed to the user of the database by interpreting (e.g. viewing or reading) the data. IC-Objects as well as IOIs can be connected by relationships, thus forming a network of information. For example, IC-Objects can be persons and IOIs can be images showing persons.

1.3.4 Presentation Level

The presentation schema describes how the information described in the content schema is presented to the user in the context of a specific application. This includes decisions about which links and thus which information are available to the user as well as entry points and guided tours through the available information.

1.4 Structure of this Thesis

This thesis is structured as follows:

Chapter 2 contains definitions of the terms "hypertext", "multimedia", and "hypermedia", as well as definitions of several other terms. The chapter also contains a brief introduction of the object model of the object oriented design notation OMT [Rumbaugh *et al.*, 1991], which is used throughout this work.

Chapter 3 describes existing hypermedia database management systems and their data models. Hypermedia capabilities of commercial database management systems are also surveyed. Chapter 3 also discusses the areas of research which are outside of the scope of this work, because discussing these areas of research helps to define the scope of the work presented here by defining the topics *not* covered by this thesis.

One of the guiding design principles of the HyperD architecture is the goal of achieving data independence. The concept of data independence and how it is addressed within existing systems is discussed in Chapter 4.

Chapter 5 describes in detail the storage level, the media level, the content level, and the presentation level of the HyperD architecture.

Chapter 6 discusses how the concept of data independence is incorporated in the HyperD architecture.

Chapter 7 introduces the concept of Corresponding Information Content and shows how the HyperD architecture can be extended to implement this concept.

Chapter 8 contains the description of a partial and prototypical implementation of the storage level and of the media level of the HyperD architecture using the commercial database management system Illustra.

The work presented in this thesis is summarized in Chapter 9.

Chapter 2

Introductory Remarks

This chapter contains definitions of important terms and concepts. In the first section, definitions for "hypertext", "multimedia", and "hypermedia" as they appear in various articles are discussed. Then some terms like node, link, and anchor are introduced, as well as some terms relating to database systems. The last section of this chapter describes the object model of the object oriented design notation OMT [Rumbaugh *et al.*, 1991], which is used throughout this thesis.

2.1 Definitions

This section takes a look at the various definitions for the terms "hypertext", "multimedia", and "hypermedia" which have been used in past. Then working definitions for these terms are proposed, which are used in the context this thesis. The next section contains definitions of terms used in the hypertext community to describe certain concepts used in hypertext and hypermedia modeling, while the last section contains definitions of some general terms.

The following quote, which is taken from [Harney *et al.*, 1991], still describes the current situation:

> The term *multimedia* means different things to different people. For some, it is the simplest combination of text and graphics on a personal computer. For others, it is the combination of text, graphics, and audio. Still others think of multimedia as including video still images. With the introduction of digital motion video, the definition of multimedia computing becomes even more clouded and confusing.

No real effort seems to have been made to give a clear definition of the terms "hypertext", "multimedia", and "hypermedia". This point is emphasized by the fact that the definitions given below—which also include statements in articles which where not labeled as definitions but can be used as such—are all the definitions which were found in several *hundred* articles dealing with various aspects of hypertext, multimedia, and hypermedia.

2.1.1 Hypertext

The idea of hypertext was first formulated by Vannevar Bush [Bush, 1945]. Bush describes a hypothetical machine called "memex" which stores information on microfilm. The information is accessed by code entered on a keyboard or by levers which turn the pages of the microfilm. Two arbitrary pages can be "connected" and whenever a user displays a page he can recall the connected pages by pressing a button. By connecting a sequence of pages a "trail" is created, mirroring the way the user looked at the information. Bush even envisioned the exchange of trails between colleagues as well as documents and encyclopedias with ready-made trails, which could be bought or received on microfilm and simply be dropped into the "memex".

The term hypertext was coined by Theodor Nelson during the 1960s [Nelson, 1967; Nelson, 1965]. In [Nelson, 1990] (cited after [Blustein, 1997]) he writes:

> By "hypertext" I mean *non-sequential writing* – text that branches and allows choices to the reader, best read at an interactive screen.

> As popularly conceived, this is a series of text chunks connected by links which offer the reader different pathways.
>
> I will not argue with this definition here, but I hope it will become clear throughout the book how much more I think hypertext can be . . .
>
> Hypertext can include sequential text, and is thus the most general form of writing. Unrestricted by sequence, in hypertext, we may create new forms of writing which better reflect the structure of what we are writing *about;* and readers, choosing a pathway, may follow their interests or current line of thought in a way heretofore considered impossible.

Some definitions of hypertext focus on a technical description:

- "Hypertext is a database technology typically characterized by small fragments of text interconnected by machine-supported links." [Raymond and Tompa, 1988]
- "The concept of hypertext is quite simple: Windows on the screen are associated with objects in a database, and links are provided between these objects, both graphically (as labelled tokens) and in the database (as pointers)." [Conklin, 1987]
- "Hypertext is a *database method*, providing a novel way of directly accessing data. This method is quite different from the traditional use of queries. At the same time, hypertext is a *representation scheme*, a kind of semantic network which mixes informal textual material with more formal and mechanized operation and processes. Finally, hypertext is an *interface modality* that features 'control buttons' (link icons) which can be arbitrarily embedded within the content material by the user. These are not separate applications of hypertext: They are metaphors for a functionality that is an essential union of all three." [Conklin, 1987]
- "A *Hypertext* is a network of information nodes connected by means of relational links." [Parunak, 1990a]. According to the authors, the previous

and the following definitions are modified versions of definitions which have been circulating for several years. "A *Hypertext System* is a configuration of hardware and software that presents a Hypertext to users and allows them to manage and access the information that it contains." [Parunak, 1990a]

A further group of definitions focuses on the way information is represented or structured in a hypertext.

- "Hypertext is text to which connections between related phrases, sentences, or paragraphs have been added." [Mayfield and Nicholas, 1993]
- "Hypertext in its essence is non-linear or non-sequential text. In a hypertext system, documents consist of a collection of *nodes* connected by directed *links*. A node by itself is similar to a piece of normal text—the links between the nodes give hypertext its non-linear aspect." [Delisle and Schwartz, 1986]
- "In essence, a hypertext system allows authors or groups of authors to link information together, create paths through a body of related material, annotate existing texts, and create notes that direct readers to either bibliographic data or the body of the referenced text". [Yankelovich *et al.*, 1988]

For the purpose of this thesis, hypertext is characterized by the following properties:

- Hypertext is a means to convey information with the aid of computers.
- This information is presented to the user in (small) units called nodes.
- Nodes can be related to other nodes.
- The relationships between nodes can carry semantic information about the relationship of the content of the nodes.

2.1.2 Multimedia

As stated above in the quote from [Harney *et al.*, 1991], there is no generally accepted definition of the term "multimedia" or "multimedia data". Yet, as will be shown in this section, some common properties are part of many definitions.

In [Gibbs and Tsichritzis, 1995] multimedia data is perceived as the unifying thread to digital multimedia. A first definition of multimedia data is derived from the definition of media data as "technologies and products giving our senses access to further form of information". More specific, multimedia data is a machine-readable representation of multimedia artifacts. A multimedia artifact is the spatial or temporal composition of artifacts from various media, where artifacts are objects produced in a particular medium, e.g. photographs, images, films, video clips, or audio recordings. Media data is the machine-readable representation of these artifacts.

Gibbs also gives a more mathematical definition of multimedia data in [Gibbs, 1991], where multimedia data is called multimedia value, based on the observation that the notion of multimedia data is based on sequences. Thus he suggests the following definition: "A *multimedia value,* $\mathtt{v}$, of data type $\mathtt{D}$, is a (finite) sequence $\mathtt{d}_i$, where the encoding and interpretation of the $\mathtt{d}_i$ are governed by $\mathtt{D}$. In particular, $\mathtt{D}$ determines how the *presentation* of $\mathtt{v}$ (the physical realization of $\mathtt{v}$, within some medium, over some time interval) can be obtained from the $\mathtt{d}_i$. Presentation of $\mathtt{v}$ takes place at a rate $\mathtt{r}_D$, the *data rate* of $\mathtt{D}$."

One group of definitions for multimedia or multimedia data is limited to an enumeration of data types which a multimedia information system is supposed to be able to deal with:

- "Multimedia data is defined as data in the form of text, sound, and images (digitized images and complex vector graphics drawings)." [Woelk *et al.*, 1986]
- "Multimedia information systems are to store, retrieve, and manage still

images, video, and voices, etc. (*multimedia data*)." [Oomoto and Tanaka, 1993]

- "Multimedia combines text and data with audio and visual information (e.g., speech, music, still images, and full-motion video)." [Liebowitz and Bland, 1994]

- "The term *multimedia* generally refers to the capture, storage, retrieval and presentation of audio and video data using computers." [Lindblad *et al.*, 1994]

Related to this group of definitions is the definition given in [Athappilly *et al.*, 1994], where the authors do not strive for an accurate definition, but attempt to have a clear notion about the basic structure of multimedia: "Essentially, multimedia computing consists of using four elements of communication technology – (i) television/video, (ii) personal computers, (iii) audio, and (iv) laser storage systems (i.e. video disc and compact disc) to bring about the synchronization of text, animation, sound and video to convey information."

A further group of definitions emphasizes the integrative aspect of multimedia, especially the temporal or spatial coordination of objects of the various media types:

- "*Multimedia* refers to the composition of multiple *monomedia* which should be synchronized *temporally* and *spatially*. In a movie, text, motion video, and sound should be synchronized so that the motion of one's lips should match his talk temporally and captions should be located spatially at an appropriate position on the screen." [Eun *et al.*, 1993]

- "Multimedia combines the interactivity of a computer with a natural user interface that includes audio, video and real images." [IBM Corporation, 1994]

- "For the purpose of this article, we offer the following working definition of multimedia computing: Multimedia combines motion and still video, special effects, synthetic video, fast graphics and text with the interactive capabilities of a personal desktop computer." [Harney *et al.*, 1991]

- "Multimedia refers to the integration of text, images, audio, and video in a variety of application environments." [Little and Ghafoor, 1993]
- "Ein Multimedia-System ist durch die rechnergesteuerte, integrierte Erzeugung, Manipulation, Darstellung, Speicherung und Kommunikation von unabhängigen Informationen gekennzeichnet, die in mindestens einem kontinuierlichen (zeitabhängigen) und einem diskreten (zeitunabhängigen) Medium kodiert sind."[1] [Steinmetz, 1993]
- "Multimedia systems combine a variety of information sources, such as voice, graphics, animation, images, audio, and fullmotion video, into a wide range of applications. The big picture shows multimedia as the merging of three industries: computing, communication, and broadcasting". [Furht, 1994]
- "Multimedia relates to the information type presented. Working with multimedia systems enables the user to create, edit, transmit, receive, store, retrieve, compute, render, and delete two or multiple types of information such as data, text, vector graphics, pixel-oriented images, video signals (moving pictures), and audio signals (voice and sound). Multimedia systems require synchronization of some of the information types in communication, processing, and presentation, and in the longer term will be characterized by the completely digital integration of all information types." [Armbrüster and Wimmer, 1992]

In the context of this thesis, we use the following definition of multimedia by Bordegoni [Bordegoni, 1992]:

> A multi-media application consists of a set of independent displayable pieces of information (text, graphics, animation, video and audio)

[1] A multimedia system is characterized by computer based, integrated creation, manipulation, presentation, storage, and communication of independent information, which are encoded in at least one continuous (time dependent) and one discrete (time independent) medium.

which have to be integrated, combined, stored and displayed. Applications have to deal with temporal, spatial and content-based aspects of multi-media information: temporal relations among media have to be identified in order to synchronize and compose in time media information; media information requires to be composed in space and displayed in some form, and finally the description of the structure of each media information has to be provided.

2.1.3 Hypermedia

There is also no generally accepted definition for the term "hypermedia". There are (at least) three possible approaches to define the concept of hypermedia:

- as a hypertext augmented with multimedia capabilities
- as a multimedia system augmented with hypertext capabilities, e.g. links
- as the merging of the (equally important) concepts of hypertext and multimedia

The first approach (hypertext augmented with multimedia capabilities) is used in the following definitions:

- "*Hypermedia* is simply an extension of hypertext that incorporates other media in addition to text." [Yankelovich *et al.*, 1988]
- "Hypermedia, as an extension of hypertext incorporating information types other than text, denotes an organizational concept of nonlinear network-like structuring of multimedia documents. It provides computer-supported nonsequential authoring and reading, i.e., interactive branching and dynamic display using information objects and multichoice links in between. Hypermedia documents enable the 'reader' to navigate along defined multiple paths through related information material, annotations, figures, film

scenes, bibliographical data, etc. within an information network of links and nodes". [Armbrüster and Wimmer, 1992]

- "A hypermedia system is an interactive multimedia system that is based on the principles of hypertext. In hypertext, text segments, or nodes, are accessed by links which function similarly to keywords. As a result, hypertext systems provide interactive instruction by essentially providing users with subject text navigation control. In hypermedia, nodes may be composed of any kind of media including video, audio, graphics, and animation." [Liebowitz and Bland, 1994]

While the second approach (multimedia augmented with hypertext capabilities) has to our knowledge not been used to define the term "hypermedia", some examples of definitions using the third approach (the merging of hypertext and multimedia) exist:

- "Hypermedia is well known as a simple and natural extension of multimedia and hypertext. Multimedia provides the richness in data types that facilitate flexibility in expressing information, while hypertext provides a control structure that supports an elegant way of navigating through this data in a content-based manner. The power of hypermedia results from merging both techniques." [Lenssen, 1994]

- [Maurer *et al.*, 1993a] define hypermedia systems as a subclass of interactive information systems with the following properties:
  - They contain various types of information ("multimedia") such as text, graphics, pictures, sounds, software-packages, etc.
  - Once a user has located a chunk of information (i.e., a primitive hypermedia "node") it is automatically "visualized" (i.e., displayed) by means of a certain procedure.
  - Once a node has been visualized there will be a number of references ("links") to other chunks of information which can be followed (by, e.g., clicking on one of the link-icons shown).

Thus, we can say that the "basic hypermedia paradigm" is multimedia information structured in accordance to the "node-link"model.

- "Hypermedia is a style of building systems for information *representation* and *management* around a network of multi-media nodes connected together by typed links." [Halasz, 1988]

A further alternative is to characterize hypermedia by the features a hypermedia system should have. For example, in [Garzotto *et al.*, 1993b] hypermedia is characterized by two major features:

1. The prevalence of multimedia data (e.g., text, pictures, graphics, video, sound, animation) over traditional, formatted data.
2. The emphasis on interactive navigation, i.e., traversing explicit connections among information.

In the same paper the terms *visual logic* and *browsing semantics* of an application are defined:

visual logic: how the data structures are organized visually and perceived by the user

browsing semantics: how navigation proceeds across the data structures

Yet another definition is given in [Akscyn *et al.*, 1988], where it is mentioned that there is no generally accepted definition of hypermedia, but that most hypermedia and hypertext systems can be characterized by the following features:

- Information is chunked into small units.
- These units are displayed one per window. This feature is obviously outdated.

- Units of information are connected by links, users traverse the database by selecting links.

- Users build information structures by creating, editing, and linking units.

- Mechanisms for concurrency control exist.

For the purpose of this thesis, we define the concept of hypermedia as the merging of the equally important concepts of hypertext and multimedia. As will be shown in Chapter 5, we use an extended node and link model not only to model the relationships between nodes—where nodes are (parts of) multimedia presentations—but also to model the temporal and spatial relationships of the monomedia used to construct the multimedia nodes. Thus we achieve a thorough interweaving of hypertext and multimedia features.

2.1.4 Technical Terms

This section contains definitions of key concepts used in hypermedia modeling, especially definitions of "node", "link", and "anchor". Note: these concepts are used to model instances and not the information usually found in the schema of a database.

In very few articles definitions for all three concepts are given. One example is a formal definition of "node", "link", and "anchor" given in [Bra and Houben, 1992][2]:

Node: "A node is described by a couple (i,v), where i is the object's identification and v is the node's value (its contents or state)."

[2]Note: the naming used in the definitions can be confusing. i is always the object identifier, n and n' as used in the definition for link are object identifiers of anchors. v is always the "value" of the object, e.g. in the case of a node v is the content of the node.

Link: "A link is described by a quadruple (i, n, n', v) where i is the object's identification, n is the link's source anchor, n' is the link's destination anchor, and v is the link's value (its 'meaning')." The "link value" can be compared to the "link type" described in Section 2.1.4.2.

Anchor: "An anchor is described by a couple (i,v), where i is the object's identification and v is the anchor's value (its location in a node)."

A very concise definition of "node", "link", and "anchor" is given in [Mayfield, 1995]:

> While the distinction between hypertext and multimedia systems has blurred in recent years, for the purposes of this chapter **hypertext** will refer to *text* that has been augmented with connections from one piece of text to another. Such connections will be called **hyperlinks,** or **links** for short. The text connected by links is often divided into small pieces, which will be called *nodes* throughout this chapter. The origin of a link is often a small sequence of words in the source node called an **anchor**.

2.1.4.1 Node

A node is the basic unit of presentation. The term originates from the hypertext community, where a node is the amount of text which is displayed at one time. Originally it was the amount of text which could be displayed on one screen, as for example in KMS [Akscyn *et al.*, 1988], later the notion was extended to allow for scrolling of text. It is often emphasized that each node should represent one idea, see for example [Conklin, 1987].

Typical definitions for "node" are:

- "An information fragment in a hypertext is called a node. Thus, hypertext is made up of a collection of distinct named information fragments. Conceptually this information fragment usually describes a single concept or topic. The names may be assigned explicitly by the user or they can be assigned automatically." [Lange, 1990]

- "A node is a chunk of information that deals with a single theme. It expresses a clear thought and is not necessarily dependent on external information for understanding of the concepts conveyed within. The node's main function is to display information which could take the form of text, graphics, sound, video, or animation". [Robertson *et al.*, 1994]

Note that all definitions above contain the idea that nodes represent one single idea or concept and thus the content of a node should be understandable on its own.

Another group of definitions emphasizes the way the content of the node is represented, e.g. which media can be used:

- "A node by itself is similar to a piece of normal text — the links between nodes give hypertext its non-linear aspects. The nodes of a hyperdocument are not restricted to be text. They can represent graphical images, combined text and graphics, digitally encoded voice, or even an animation." [Delisle and Schwartz, 1986]
- The system NoteCards [Halasz and Schwartz, 1994] uses the term "notecard" instead of "node": "A notecard is an electronic generalization of the 3×5 paper notecard. Each notecard contains an arbitrary amount of some editable substance such as a piece of text, a structured drawing, or a bitmap image. Each card also has a title."

Another possibility to define the concept of a "node" is by describing it from the following three perspectives [Parunak, 1990b]:

Node content: e.g. text, images, animation, audio, or even programs [Smith and Weiss, 1988] plus a definition of location for each type of node, e.g. for text the first and the last character of a region of text.

Node typing: The type of a node refers to the way the content of a node is used, for example, the content of a node is an "argument" or a "position". This usage of the term "type" is different from the usage of "type" in the area of data modeling: while a type in data modeling describes the structure of an entity to be modeled, the type of a node is merely a label attached to the node. Nodes which are typed in this manner can then be connected by typed links (see next section) to form a semantic network of links and nodes. For example, a node of type "argument" has a link typed "supports" to a node of type "position". Again, the type of a link is only a label attached to the link. This kind of typing does *not* impose any restrictions on which types of links can be connected with which types of nodes.

Node structure: defines the inner structure of the node by identifying higher order objects in the node, for example words for text or musical phrases in audio recordings. These inner structures can be used as anchors (see the next section) for links.

2.1.4.2 Link

A link is a connection between two or more nodes.

Typical definitions are:

- "A connection between two nodes is called a link. When a link is activated, say by a mouse click, one can jump to the node the link points to." [Lange, 1990]
- "*Links* are connections between anchors. They are directional in that they have a source and a destination anchor." [Riley, 1990]

One further important concept is "anchor". An anchor is a node or an arbitrary region within a node which is the start or the end of a link:

- "*Anchors* are the locations in documents to which links are attached." [Riley, 1990]
- "The anchors, or endpoints, of a link are its origin and its destination." [Parunak, 1990b]

One group of definitions emphasizes that links model a specific relationship between the concepts expressed in nodes:

- "Anchors and links are objects which bind together associated concepts expressed within an information space. By performing some action on an anchor point in the media the user traverses a link to some other related concept in the information space." [Robertson *et al.*, 1994]
- This is the definition for link which is used in the NoteCards system [Halasz and Schwartz, 1994]. Again, the term "notecard" is used instead of "node": "Links are used to interconnect individual notecards into networks or structures of related cards. Each link is a typed, directional connection between a source card and a destination card. The type of a link is a user-chosen label specifying the nature of the relationship being represented."

The concept of a "link" can also be described by discussing its properties [Parunak, 1990b]:

directionality: A link can be bidirectional (undirected) or unidirectional. A link is unidirectional if traversing the link in one direction is different from traversing the link in the other direction. For example, because for traversing the "other" direction a different operation has to be invoked.

topology: describes restrictions on links. For example, links can be restricted to form linear or hierarchical structures or directed acyclical graphs.

link types: Links can be typed to gain better rhetorical capabilities, for an example see the preceeding section on nodes. Again, the term "type" is used in the same manner as in "typed nodes" (see above for a discussion).

link anchors: The anchors of a link can either be whole nodes, a higher order object within a node, or simply a certain point or region.

link modes: A link can be fixed or virtual (i.e. computed at runtime).

Marshall [Marshall, 1990] analyzes analytical activities and deduces the following requirements for links:

- Links are named, typed, and have direction. Names specify the kind of relationship (supports, refutes, ...), type specifies the behavior of the link and direction is used to express relationships between nodes or their contents. Note: this is yet another definition for "type". In the paper no examples for link behaviors are given. Possible examples are: pop-up (the content of the target node is shown in a new window), replace (the content of the target node is shown instead of the content of the source node), or insert (the content of the target node is inserted into the content of the source node at the position of the anchor).

- Links are n-ary, i.e. links can start or end at more than one node. N-ary links are useful to integrate information in more than one node. It depends on the hypermedia system which actions are caused by selecting a n-ary link. The user could either be presented with a menu to select one of the destinations or all destinations could be displayed in parallel.

- Links can either connect nodes or refer to nodes. Reference links are components within a node that contain a name or an address that refers to another node (or a region within another node), or a procedure for retrieving that node; thus a link's destination can be computed at traversal. A connection link is a fixed connection from one node to one particular node (or a group of such particular nodes).

- Links can be anchored in a span of text. The extent of an anchor on either end of a link can range from a single point (e.g. NoteCards [Halasz and Schwartz, 1994]) to an arbitrary range of text or graphics (span-to-span linking, used e.g. in Intermedia [Yankelovich *et al.*, 1988]) to entire nodes only (one example is gIBIS [Conklin and Begeman, 1988]). Span-to-span linking provides a fine granularity and can be used to relate specific information found in nodes.

- Links are marked to reflect their properties. This refers to the way links, represented through their anchors, are presented to the user. The user should be able to detect the presence and absence of anchors without additional actions, determine the range of the anchor, and distinguish between different types of links.

- Links can be annotated.

2.1.4.3 Further Hypermedia Definitions

This section contains further definitions of important hypermedia concepts.

Web: This term is often used to denote a network comprising nodes and the links connecting them. Sometimes the term "hyper-web" is used.

Index: In hypertext modeling, an index is a page which allows to access all (important) pages of a web. It can be compared to a "table of content" page or an index as usually found in the back of a book.

Guided Tour: A definition of this important concept in hypertext/hypermedia modeling, which is sometimes also called "path", is given in [Parunak, 1990b]: "A bare network of links and nodes is well-suited to random browsing, but many applications of hypermedia presuppose a basic trajectory through the hyperbase, with the rest of the material available as needed. Paths support such applications by giving writers a way to define a backbone that readers should follow, and to which they can readily return after any digression. Topologically, the path imposes a linear topology on a much more complicated network, thus combining the cognitive advantages of the simpler topology with the flexibility of the more complex one."

Shot: A shot is a sequence of images in a video which depicts the same scene, signifies a single camera operation, contains a distinct event or an action like for example significant presence and persistence of an object, or which is chosen as a single indexable entity by the user [Flickner *et al.*, 1995]. Shots are often used to structure videos after insertion in the database. The automatic detection of shot boundaries (shot detection) is investigated e.g. in [Christel, 1995; Flickner *et al.*, 1995].

2.1.5 General definitions

This section contains some definitions which are not particular to the area of hypermedia research, but are important for the overall understanding of this thesis.

Data: "By **data**, we mean known facts that can be recorded and that have implicit meaning." [Elmasri and Navathe, 1994, p. 2]

Database: "A **database** is a collection of related data" [Elmasri and Navathe, 1994, p. 2]

Elmasri and Navathe also describe the following implicit properties of databases [Elmasri and Navathe, 1994, p. 2]:

- "A database represents some aspects of the real world, sometimes called the **miniworld** or the **Universe of Discourse (UoD)**. Changes in the miniworld are reflected in the database."
- "A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database."

- "A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested."

Database Management System: "A **database management system (DB-MS)** is a collection of programs that enables users to create and maintain a database. The DBMS is hence a *general-purpose* software system that facilitates the process of defining, constructing, and manipulating databases for various applications." [Elmasri and Navathe, 1994, p. 2f]

According to [Ullman, 1988, p. 2], database management systems have to have two qualities:

1. The ability to manage persistent data, and
2. The ability to access large amounts of data efficiently.

Database System: "A database system is the combination of a database and the corresponding database management system." [Elmasri and Navathe, 1994, p. 2f]

Data Model: "A **data model** is a set of concepts that can be used to describe the structure of a database. By *structure of a database,* we mean the data types, relationships, and constraints that should hold for the data. Most data models also include a set of **basic operations** for specifying retrievals and updates on the database." [Elmasri and Navathe, 1994, p. 23f]

According to [Ullman, 1988] a data model is a mathematical formalism with two parts:

1. A notation for describing data
2. A set of operations used to manipulate that data

In [Silberschatz *et al.*, 1996] a data model is "a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints."

2.2 Notation

This section contains a brief introduction to the notation of the object model of the Object Modeling Technique (OMT) [Rumbaugh *et al.*, 1991]. This notation is used throughout the thesis for the description of the architecture as well as for the examples.

This section is not intended as an introduction to object orientation or object oriented design, therefore concepts like "class" or "object" are not formally defined.

2.2.1 Classes

Figure 2.1 shows the full notation for classes used in OMT and two abbreviated notations. On the lefthand side the full notation is shown: A class is represented by a rectangle with two horizontal lines, forming three sections.

- The top section contains the name of the class (`ClassA`).
- The mid section contains the description of the attributes of the class. The description consists of a mandatory attribute name (e.g. `attributeA` or `attributeB`) and an optional data type (`dataType`). An attribute with a preceeding $ (e.g. `$attributeC`) is a class attribute. An attribute with a preceeding `/` (e.g. `/attributeD`) is a derived attribute. All other attributes are instance attributes.
- The bottom section contains the description of the methods of a class. In this thesis the term "method" is used instead of the term "operation" as used in [Rumbaugh *et al.*, 1991]. The description consists of a mandatory method name (e.g. `methodE` or `methodF`), an optional argument list, and an optional return type. A method with a preceeding $ (e.g. `$methodG`) is a class method.

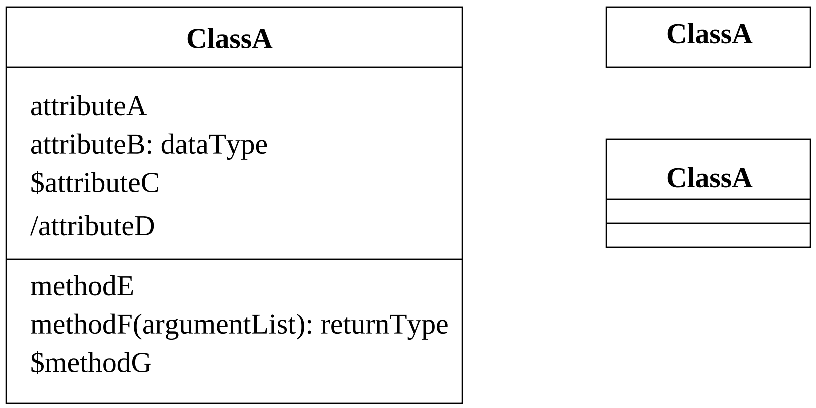

Figure 2.1: OMT: Classes

The righthand side of Figure 2.1 shows two abbreviated notations for classes. On the top the abbreviated notation as used in OMT is shown, consisting of the name of the class in a rectangle. Below an alternative notation is shown: a rectangle with the class name and two empty sections. This notation does *not* mean that the class does not have attributes or operations. This notation is used because one of the drawing tools used to create the figures does not support the original notation.

2.2.2 Objects

Figure 2.2 shows the notation for objects. An object is denoted by a rectangle with rounded corners and the name of its class in parenthesis. For example, the object shown in Figure 2.2 is an instance of class `ClassA`. Optionally, the graphical notation of an instance can also contain the values of its attributes as it is shown on the righthand side.

At the bottom of Figure 2.2 a class and one of its instances is shown. The dotted arrow is the notation for an instantiation relationship.

In the examples of chapter 5 the class an object is an instance of is either

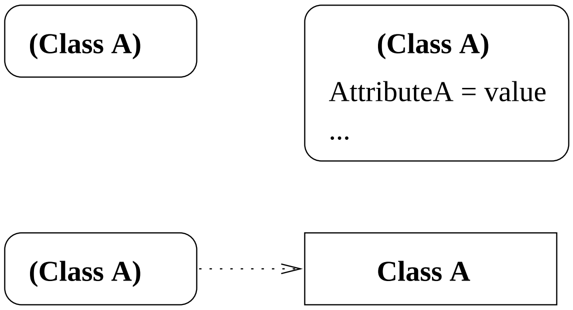

Figure 2.2: OMT: Objects

drawn explicitly and connected to the object with an instance-of relationship, or the name of the class is given as the first part of the name of the object (see for example Figure 5.6).

2.2.3 Relationships

Throughout the thesis the term "relationship" is used instead of "association" as used in [Rumbaugh *et al.*, 1991]. Instances of relationships are referred to as "relationship occurrences".

Figure 2.3 shows various notations for relationships. From top to bottom, the notations shown are used to denote:

- A regular relationship between two classes, expressed by a line connecting the rectangles representing the classes. A relationship has an optional name (`relationshipName`). Also the role each of the classes has in the relationship can be shown: `roleA` as the role `ClassA` has in the relationship, `roleB` as the role `ClassB` has.
- Relationships can also be n-ary, here a ternary relationship between classes `ClassC`, `ClassD`, and `ClassE` is shown.
- The third relationship from the top has a class attached to the relationship.

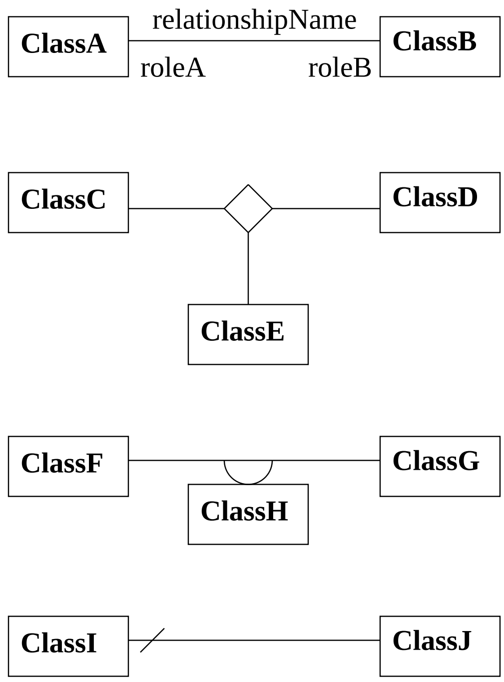

Figure 2.3: OMT: Relationships

This is the notation for relationship classes. In this case the relationship class `ClassH` relates the (independent) classes `ClassF` and `ClassG`.

- The relationship between `ClassI` and `ClassJ` is derived, shown by the short line crossing the line denoting the relationship at an angle. The relationship is derived either from other relationships or from values of attributes and does not add additional information to the diagram, but is used for clarification.

Figure 2.4 shows the notations to express the multiplicity of relationships. From top to bottom, the notations express:

- Each instance of class `ClassA` takes part in exactly one occurrence of the depicted relationship.

- Each instance of class `ClassB` takes part in zero, one, or many occurrences

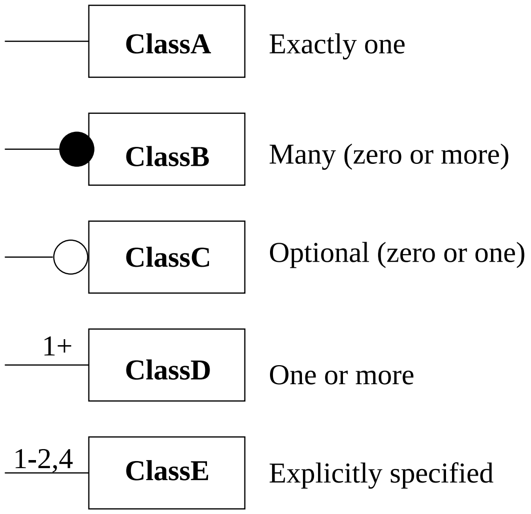

Figure 2.4: OMT: Multiplicity

of the depicted relationship.

- Each instance of class `ClassC` takes part in either one occurrence of the depicted relationship or in no occurrence at all.

- Each instance of class `ClassD` takes part in one or many occurrences of the depicted relationship.

- The multiplicity can also be given explicitly. For example, each instance of class `ClassE` takes part in one, two, or four occurrences of the depicted relationship.

Figure 2.5 shows an example: `InstanceA1`, `InstanceA2`, and `InstanceA3` are instances of `ClassA`; `InstanceB1` is an instance of `ClassB`. Also shown are three relationship occurrences between `InstanceB1` and `InstanceA1`, `InstanceA2`, and `InstanceA3`. These relationship occurrences are instances of the relationship between `ClassA` and `ClassB`. The instance-of relationships are also shown in the figure.

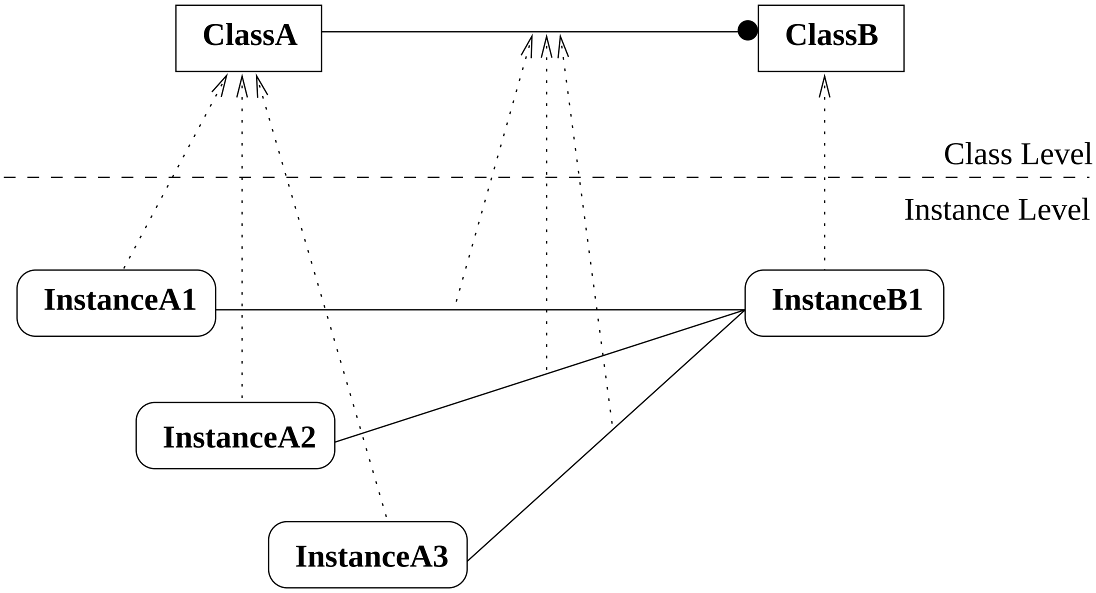

Figure 2.5: OMT: Example

2.2.4 Aggregation and Inheritance Relationships

This section describes the notation for two special classes of relationships: aggregation relationships and inheritance relationships. Note: While aggregation relationships are defined between classes, they exist between objects. Inheritance relationships are defined and exist between classes.

The notation for aggregation relationships is shown in Figure 2.6. The two notations shown are semantically equivalent. Both express that class `AssemblyClass` is composed of or can be decomposed into parts `PartClassA` and `PartClassB`.

The notation for inheritance relationships is shown in Figure 2.7. On the left-hand side the notation for single inheritance is shown: class `SuperClass` is the superclass of the subclasses `SubClassA` and `SubClassB`. Instances of a subclass are a subset of the set of instances of the superclass and inherit all attributes and methods from the superclass. The notation for multiple inheritance is

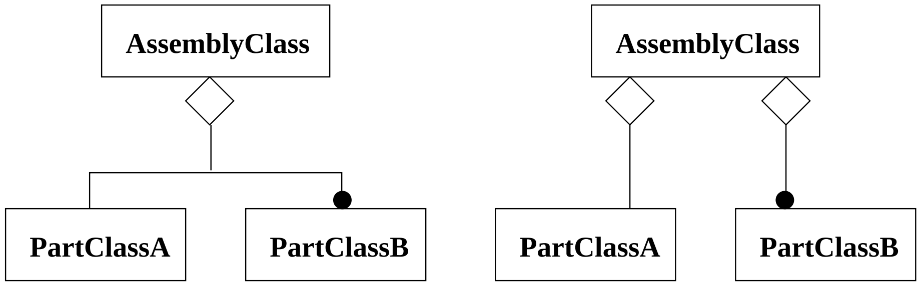

Figure 2.6: OMT: Aggregation

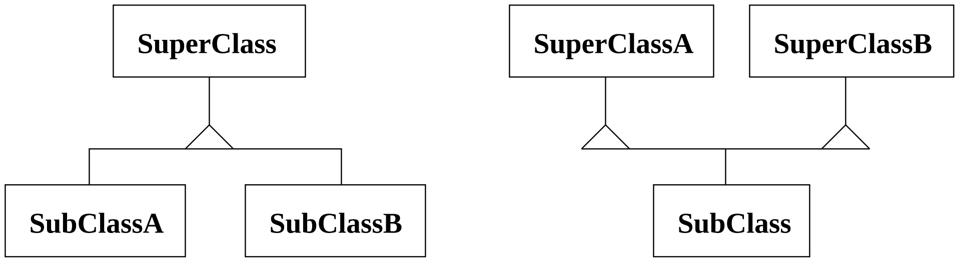

Figure 2.7: OMT: Inheritance

shown on the righthand side: class `SubClass` is the subclass of the superclasses `SuperClassA` and `SuperClassB`.

Subclassing of relationships is shown as an is-a relationship drawn with dotted lines as suggested in [Rumbaugh, 1993]. For an example, see Figure 2.8.

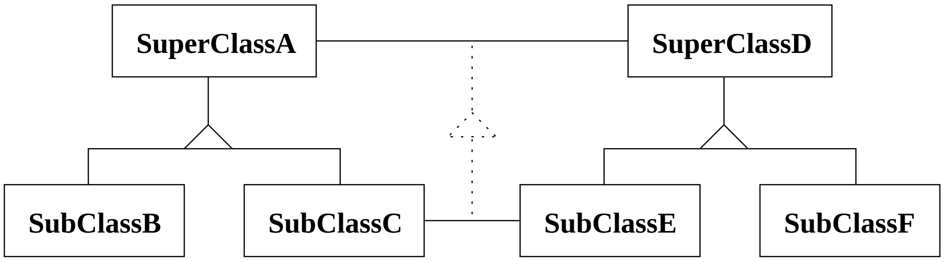

Figure 2.8: OMT: Relationship Inheritance

Chapter 3

Background and Related Work

This chapter gives a survey of related research areas and previous work. Related research areas are those areas which are outside of the scope of this thesis, but provide either the technological basis for the HyperD architecture or facilitate a deeper understanding of the information as it is represented in the HyperD architecture.

The chapter also contains an overview of previous research which is similar to certain aspects of the HyperD architecture and discusses the hypermedia capabilities of commercial database management systems.

3.1 Related Areas

The research described in this section helps to define the scope of the work presented here by explaining topics related to but not covered by this thesis. It is also shown how the work presented in this section can be embedded into the HyperD architecture. This section is not intended as a comprehensive survey, but gives an overview by focussing on important concepts and selected research.

3.1.1 Media Server

Media server provide the technological basis for centralized multimedia storage, processing, and playout, often in the context of video-on-demand. Multimedia storage servers manage continuous media, the emphasis is placed on ensuring continuous retrieval of a media stream from disk. [Berra *et al.*, 1992] gives an overview and an overall assessment of the state-of-the art in multimedia information technology. Issues covered include multimedia applications, multimedia data management, multimedia networking, and distributed systems.

In this section, we follow the approach taken in [Gemmell, 1995], where an overview of multimedia storage servers is given, and describe two important fields of research for media server:

- Continuous media recording and retrieval, which deals with the process of continuous media playback and the management of a media server.

- Managing digital multimedia storage, which deals with problems regarding the placement of data blocks. Data blocks can occupy several physical disk blocks, either on disk or on tertiary storage.

For each of these areas first the general concepts are introduced and then selected research results are presented. The section ends with a discussion of the relationship of the area of research concerning media server and the results presented in this thesis.

Further work in this area is presented e.g. in the Proceedings of the ACM International Multimedia Conference [ACM, 1993; ACM, 1994; ACM, 1995] and in the Journals *ACM Multimedia Systems* and *IEEE MultiMedia*.

3.1.1.1 Research Issues for Continuous Media Recording and Retrieval

This section describes important research issues in the area of continuous media recording and retrieval.

Single-stream playback: Media blocks need to be pre-fetched and buffered after reading from disk for consumption by the playback process, because feeding the playback process from disk directly results in bursty playback. In practice this has been found to be necessary even for modern disks, whose throughput is higher than the consumption rate of the playback process. The challenge of single-stream playback is to prevent starvation of the playback process while minimizing the buffer space required and minimizing the initiation latency, i.e. the time to fill the buffer sufficiently before the playback process can start.

Multistream retrieval: In practice, media servers are required to process many requests for different files or different parts of the same file at the same time. The goal is to serve the highest possible number of requests while meeting the continuous playback requirements of all existing streams, i.e. no stream starves.

Disk scheduling: Disk scheduling deals with a class of algorithms which schedule the reading of media blocks for different streams from a disk in a timely manner. The algorithms try to reduce seek time and rotational latency to achieve high throughput or to provide fair access for each stream.

Round length and latency trade-offs: Processing requests in rounds means that during each round the server retrieves a media block sequence of arbitrary length, which can also be zero, for each stream. Latency is the time between two reads for the same media stream, which is not constant for some algorithms. The choice of these parameters affects the buffer requirements and also the playback initiation delay.

Reading and buffering requirements: The scheduling algorithms presented so far try to ensure that data production (i.e. reading data from disk) never lags behind data consumption, this property is referred to as "buffer-conserving". Different strategies to keep the buffers from draining can be used.

Managing buffers for maximum consumption: Another problem is how to cope with buffers which have not yet been fully used when the next read is performed. The data read from disk does not fit into the buffer, thus—at least partially—wasting the time spent to read the data. Solutions proposed include up to three buffers per stream.

Admission control: Admission control algorithms determine if a new stream with a certain quality-of-service requirement can be served in addition to the already existing streams. Not every stream has to be served perfectly all the time, so quality-of-service requirements are used to define "how good" the stream has to be served:

- **Deterministic:** The stream has to be served without any loss of data and without letting the stream starve.
- **Statistical:** The stream is guaranteed to be served with a certain probability, e.g. the stream is served 95 percent of the time.
- **Background:** No guarantees are given, the stream is served only if the server-workload permits.

Dealing with service guarantees and deadlines: If the server is overloaded and cannot fully serve all streams, a graceful degradation of service for certain streams has to be accomplished. This can be done e.g. by dropping media blocks or by using decompression schemes which can be scaled to decode a lower resolution output.

3.1.1.2 Selected Research Results for Continuous Media Recording and Retrieval

This section describes selected research results in the area of continuous media recording and retrieval.

In [Gemmell and Christodoulakis, 1992] fundamental principles for the retrieval and storage of delay-sensitive multimedia data, e.g. digital audio, animations, and video, are explored. A general theoretical framework for studying the performance of any playback system is developed and used to discuss questions like consumption rate vs. block size, storage placement strategies, buffered systems, and multichannel playback. The main principle explored in the article is "Minimizing Buffer Utilization and Start Time".

[Staehli *et al.*, 1995] describe specification techniques for quality of service (QoS) requirements. QoS requirements describe in which range a presentation can adapt to the available communications bandwidth and the output device resolution. The proposed methodology for QoS specification contains the following steps:

1. Definition of an ideal presentation: specifies device-independently when and how a certain media stream has to be presented.
2. Choosing of an error interpretation: compares the ideal presentation to the actual presentation, which deviates because of device limitations, choice of presentation algorithm, or choice of scheduling policy.
3. Constructing a user model: a user model judges the suitability of a presentation for a certain application based on the error interpretation.

[Ozden *et al.*, 1996] deal with storage and playback of continuous media (audio and video), placement of this data on disk-arrays, admission control, and timely playback to the client with block pre-fetching. This approach can also cope with disk failures: parity groups are used to reconstruct missing blocks which are not available because of disk-errors.

Similar capabilities are incorporated in AMOS (Active Media Object Stores) [Thimm and Klas, 1995]. The playout management of AMOS deals with the management of output devices, with the management of the transfer of data streams to the output devices, and with the enforcement of synchronization within as well as between media streams. AMOS also supports interactivity in the form of VCR-like controls and provisions for smooth responses to these requests by prospective buffering and adaption of the media stream.

3.1.1.3 Research Issues for Managing Digital Multimedia Storage

This section describes important research issues in the area of managing digital multimedia storage.

Placing data blocks for optimal service: A file can be stored either contiguously or scattered. A contiguous file can be implemented easily, but this can lead to fragmentation and to large overheads during insertion and deletion. Therefore contiguous files are used for read-only systems, e.g. video on demand.

For continuous media the following kinds of placement strategies are distinguished:

Contiguous placement: One further advantage of contiguous placement for continuous media, in addition to easy implementation, is the low intrafile seek time: when reading a number of blocks only one seek is necessary while in a scattered file in the worst case one seek for each block is necessary.

Constrained placement: Here the number of intrafile seeks for a given number of blocks is limited, thus reducing the disadvantage of scattered files described above.

Log-structure placement: With this approach, modified blocks are not stored on their original position but in a special area, thus saving intrafile seeks during stream writing. This approach is beneficial for systems with extensive recording/editing, but more complex to implement and not suitable for read-only applications, because due to the non contiguous storage of the media streams additional seek operations are necessary.

Data striping and data interleaving: A multimedia file can be stored non-redundant on several disks to allow more concurrent accesses than the throughput of one disk permits. With the *data striping* approach, all disks move in sync so that the same physical sector of each disk can be read simultaneously. This allows to increase the transfer rate by the number of disk drives used. When using the *data interleaving* approach, the disks operate independently, for each round and stream either blocks from all disks or from one disk are retrieved. Data striping and data interleaving can be combined to build scalable video server which can serve multiple streams of the same video.

Tertiary storage and hierarchies: For large scale servers tertiary storage devices with high mount time and, compared to magnetic disks, low cost per gigabyte have to be used for economic reasons. Such tertiary storage devices are e.g. optical disks or magnetic tapes, which are operated by juke-box like devices or moved by robotic arms to reading devices. Several approaches to decrease the startup time or the overall availability are investigated.

3.1.1.4 Selected Research Results for Managing Digital Multimedia Storage

This section describes the work done by Gibbs *et al.* as one typical of example of the results in this area of research as well as other typical examples.

In [Gibbs, 1991; de Mey *et al.*, 1992; Gibbs *et al.*, 1992; Gibbs *et al.*, 1993b; Breiteneder *et al.*, 1992b; Gibbs *et al.*, 1994; Gibbs and Tsichritzis, 1995] the authors describe an object-oriented framework for composite multimedia, based on active objects implemented as real-time processes. In this context, “A multimedia object is considered to be an ‘active’ object which produces and/or consumes multimedia values. The values have to be of a specified type and are produced or consumed at their associated data rates.”. The term “multimedia value” can be compared to the now more popular term “stream”. Three types of media objects are distinguished: sources, which generate or retrieve multimedia values, filters, which process multimedia values, and sinks, which store or display multimedia values. Media objects can be hardware or software components, their specific properties are hidden in the framework.

Time dependent values are mapped from their natural time to the time needed by the application. The mapping is used to map a media value to temporal coordinates for temporal synchronization. This also allows to incorporate interactive operations like cue, start, and stop for all time-dependent media objects and composites built of these objects. The framework defines "quality factors", e.g. voice-quality, FM-quality, or CD-quality for audio, which describe the requirements of applications without giving explicit references to physical properties like sample size, which are hidden inside the framework.

The emphasis of the work is placed on modeling the *technical process* of multimedia retrieval, processing, and display, as well as incorporating different types of hardware into an object-oriented framework. Gibbs *et al.* also describe several applications of their framework, e.g. the "virtual museum" or "video actors" [Breiteneder *et al.*, 1992a; de Mey and Gibbs, 1993; Gibbs *et al.*, 1993a].

One example for the large number of approaches for data placement for continuous media which have been proposed, is the work described in [Kwon and Lee, 1995]: a restricted round-robin approach is used to disperse video segments onto distributed disks to provide concurrent retrieval of different portions of the same video.

One interesting approach for the integration of tertiary storage is described in [Narayana *et al.*, 1995], where a market-based economic model, which incorporates different economic paradigms like client-consultant or landlord-tenant, are used to handle policies of data caching, replication and migration of multimedia objects in the context of a large network.

3.1.1.5 Discussion

The research presented in this section provides important base technology, which can be incorporated in the HyperD architecture, predominantly in the storage level.

Important differences to our work are:

- Many of these systems only deal with one type of media—audio and video is often interlinked and is transmitted together—usually of a temporal nature. The HyperD architecture deals with arbitrary media types, both temporal and nontemporal.

- Some systems provide basic support for temporal composition, but no semantically richer relationships between nodes are supported. The HyperD architecture supports a rich set of relationships between various nodes, including temporal and spatial composition, temporal and spatial decomposition, as well as relationships with user defined meaning between arbitrary nodes.

- The systems presented in this section primarily deal with technical aspects, the HyperD architecture focusses on the semantic modeling of the hypermedia data.

3.1.2 Information Retrieval

In contrast to retrieval based on attributes associated with objects, information retrieval deals with retrieval of data based on the content of the data itself. The systems presented in this section do not elicit the kind of content which is modeled in the HyperD architecture. As noted in Section 1.3.3, in the context of the HyperD architecture the content of a piece of multimedia data is the meaning conveyed to the user when she reads, or listens to, or looks at the data.

The kind of content elicited by the systems surveyed in this section can be stored in the content level as an automatically defined default structuring of multimedia data. For example, systems like [Flickner *et al.*, 1995] detect shot boundaries, which can be used as a generic way to structure a video stream. The data model of the content level is flexible enough to describe the results generated by these systems and thus can be used as a unified interface to these systems.

As in the section on media server, we first describe important research issues in the area of information retrieval. Then selected research results are presented, before the relationship between this area of research and the research presented in this thesis is discussed.

Further work in this area of research is presented e.g. in the Proceedings of the ACM International Multimedia Conference [ACM, 1993; ACM, 1994; ACM, 1995] and in the Journals *ACM Multimedia Systems* and *IEEE MultiMedia*.

3.1.2.1 Research Issues for Information Retrieval

This section describes important research issues in the area of information retrieval, with special emphasis on content based retrieval, because this area of information retrieval has the greatest significance for the research presented in this thesis.

[Gudivada and Raghavan, 1995] give an overview of content-based image retrieval (CBIR). The "content" dealt with by this class of applications can be described by the following properties: color, texture, sketch, shape, volume, spatial relationships, objective attributes, subjective attributes, motion, text, and domain concepts. Note: this notion of content does *not* include the content of an image as perceived by a person ("The picture contains a little yellow flower and a red brick wall"). For audio, properties like spectral properties, frequency, and amplitude are used, but not the meaning of the words spoken on the recording. This type of content is only covered if it has been entered manually into a text-attribute or as a domain concept. The same article also defines important research areas for content-based image retrieval:

Data modeling: A data model for CBIR has to have a rich set of modeling constructs to capture the content as described above. Current data models are not comprehensive enough for domain-independent CBIR.

Logical features and algorithms: The data model should be able to model domain-independent and storage-efficient logical features. Efficient and robust algorithms based on these features for processing interactive queries have to be provided.

Query-specification scheme: Different query specification techniques should be used for domain experts and casual/naive users. Research areas include graphical interfaces, natural language queries, information filtering, and user modeling.

Similarity-based retrieval: While database systems typically process queries based on exact matches, CBIR needs the ability to rank images based on similarity with a given image (similarity-based retrieval). One important research topic is the development of a theoretical framework for similarity-based retrieval.

Access structures for logical features: Access schemes for CBIR act as a filter between secondary/tertiary storage and the query processing algorithms. Only those images which have a potential for being relevant pass this filter and are processed by the query processing algorithms. Examples for existing access structures are spatial access methods, which are restricted to features which can be represented numerically. To cover the full range of relevant features, more general schemes are needed.

Eliciting relevance feedback and improving retrieval effectiveness: Some attributes as well as the perceived value of images are subjective, so the user has to provide feedback about how closely certain images match the desired results. Query interfaces based on knowledge-based systems can guide the user through iterations of the query specification-feedback cycle.

Distributed Aspects: According to [Gudivada and Raghavan, 1995] the aspect of network-transparent access to distributed image collections does not seem to have been investigated, although image repositories are likely to evolve in a decentralized fashion.

Tools for feature extraction and semantics capture: While some progress on research on automated tools for extracting primitive features and semi-automated tools for capturing logical features is made (see e.g. some of the examples mentioned below), more comprehensive tools are yet to emerge.

Performance and usability issues: In this context, the CBIR community distinguishes three aspects, for which meaningful metrics should be developed:

Effectiveness measures the relevance perceived by the user of the retrieved images.

Efficiency measures the system responsiveness and interactivity.

Usability methods evaluate human performance and preference in using CBIR systems.

For a similar survey on text retrieval see [Baeza-Yates and Navarro, 1996].

3.1.2.2 Selected Research Results for Information Retrieval

This section discusses selected research in the area of information retrieval.

Chabot [Ogle and Stonebraker, 1995]: Images are retrieved from a relational database based on (manually entered) attributes, color, and shape.

[Bach *et al.*, 1993]: The article presents a general architecture for visual information management systems (VIMS) which combines image databases and computer vision systems. The architecture allows query-by-example for images. A VIMS for retrieval of faces is presented as a demonstration for the architecture.

Piction [Srihari, 1995]: Human faces in newspaper photographs are identified based on the information contained in the associated captions, i.e. regions automatically recognized as faces are related to the names found in the caption according to a constraint graph derived from the caption: a typical caption contains e.g. "From left to right: Thomas Prückler, Michael Schrefl", so the leftmost face recognized has to be Thomas Prückler.

MARIE [Guglielmo and Rowe, 1996]: The MARIE system also parses captions of photos and uses this information together with a domain specific type hierarchy structure to enhance natural language retrieval.

GRADI [Keim and Lum, 1993]: GRADI supports incremental query specification of natural language queries. The content which can be queried is either contained in the (manually entered) captions of media objects or the physical properties of media objects, e.g. size or resolution.

QBIC [Flickner *et al.*, 1995]: This prototype allows queries based on example images, user-constructed sketches and drawings, color and texture patterns, camera and object motion, and other graphical information. Some support for video is provided by segmenting videos into shots and defining representative frames (r-frames), which can in turn be queried like images. Some information about motion of objects in the video is also stored and can be queried.

PICQUERY$^+$ [Cardenas *et al.*, 1993]: This work is an example for a high-level domain-independent query language for pictorial and alphanumeric database management systems. The language allows to specify arithmetic, temporal, object evolutionary events, and content based queries tailored to a specific knowledge domain, for example a medical database about bones.

VIMSYS [Gupta *et al.*, 1991]: The VIMSYS data model describes a layered data model for the storage of images. The model provides sophisticated support for the storage of information derived directly from the image and of information entered manually as well as support for queries based on these informations. The system allows incremental retrieval of images based on similarity and domain knowledge.

HIRMA [Basili *et al.*, 1995]: HIRMA is an integrated environment supported by the lexical acquisition system ARIOSTO. HIRMA allows to query any full-text document base system by natural language sentences to obtain a document set relevant to the query. HIRMA supports hypertextual information retrieval by browsing through the document sets as well as within each document.

ForEyes [Picard, 1995]: ForEyes uses a nonlinear learning algorithm to perform queries which take subjective expectations of users into account. Users give positive and negative examples of desired results and then give feedback on the suitability of the images returned by the system. Queries are processed over features computed directly from images instead of human supplied labels.

[Simonnot and Smaïl, 1995]: This research extends the popular keyword indexing model for videos and still images to facets, which are groups of attribute/value pairs. The selection of certain attribute/value pairs for a certain facet is governed by a domain specific viewpoint. Facets can be grouped according to the kind of knowledge they represent, for example general factual knowledge (time, place, genre), media specific knowledge (focusing, lighting), specific application domains (TV news, historic reporting), content description (from the point of view of the observer of the image or video), and connotation (subjective description from the viewpoint of the observer of the video or image, e.g. cheerful, sad, tense). This approach on knowledge representation is used in combination with an information retrieval process with relevance feedback.

While this representation of "content" goes beyond the content as described in the previous section, it lacks the expressive power of IC-objects, especially because no relationships between the facets can be modeled. Also, even if facets can be grouped, this is no substitute for the expressive power that is provided by an object oriented modeling approach using IC-objects.

[Leung and Zheng, 1995] describes a data model for the systematic representation of image content. The data model is based on facts, which attach a certain meaning to an image or represent relationships between objects shown in the image. The facts are used to construct a content-based index using a relational database structure. Facts include elementary facts (the image contains a certain object, e.g. a flower), modified facts (a descriptive property of an object, e.g. a *red* flower), outline facts (a condensed description, e.g. celebration, explosion), binary facts (represent a specific interaction between facts, e.g. girl picks flower), and ternary facts (represent a particular interaction between facts; e.g. girl puts flower into basket).

The same points which were discussed above for the work presented in [Simonnot and Smaïl, 1995] are also valid here: while this representation of the image content is again more powerful than the representation of content as described in the previous section, it is less powerful than the capabilities of the data model of the HyperD architecture for the same reasons as described above.

CAFIIR [Wu *et al.*, 1994; Wu and Narasimhalu, 1994]: The CAFIIR system stores and manages facial images and criminal records. The facial images are indexed using a hierarchical self-organizing neural network. Retrieval methods include visual browsing, similarity retrieval, free text retrieval, and fuzzy retrieval.

[O'Connell *et al.*, 1996]: The article describes a multimedia object manager for massively parallel architectures, which allows content based retrieval of a vast number of multimedia objects. A variable number of agents search the database for objects which fit the description given in the agents script. User-defined-functions can be used to extract information from the multimedia objects.

[Chaudhuri and Garavano, 1996]: The authors argue that users do not query multimedia database management systems expecting exact matches, but expect a grade match. The paper distinguishes filter conditions, i.e. a value has to be above a certain threshold, and ranking expressions, i.e. how well a criterion is met. A SQL-like syntax as well as the optimization of filter and ranking expressions is described. The work only deals with text and images (color histograms).

3.1.2.3 Discussion

Important differences to our architecture are:

- Most systems deal with only one kind of media data, usually with images. While some systems also provide simple support for text, no comprehensive, media independent approach is described.
- These systems deal with a (at least potentially) large number of isolated multimedia objects. The HyperD architecture emphasizes the relationships between the objects.
- In contrast to the systems presented in this section, the HyperD architecture supports access based on navigation. The HyperD architecture features no specific support for content based queries, but these capabilities can be added easily due to the powerful modeling capabilities and due to the easy extendibility.

The results of some of these systems can be used to populate the content level of the HyperD architecture, e.g. Piction [Srihari, 1995].

3.1.3 Document Data Models/Interchange Standards

This section describes document data models and hypermedia interchange standards. The goal of this class of data models is (a) to represent structured documents containing hypermedia data and (b) to allow the easy exchange of these documents of between different hypermedia systems.

3.1.3.1 SGML/HyTime

The SGML/HyTime [Newcomb, 1995; Newcomb, 1995; Newcomb *et al.*, 1991] standard is an extension of SGML. SGML/HyTime is intended to be the international standard for representing hypertext links, and for synchronization of static and time-based information.

SGML defines a language for describing the logical structure of a text document. The language uses standardized markup to specify which segments of a document belong to which logical component of the document, e.g. which part of the document is the abstract or which part is the bibliography.

SGML specifies only logical relationships between the components of a document, but does not specify layout information, presentation information, or how to handle images or graphics. SGML also does not support time dependent data or references to destinations outside of the current document. For this reason, HyTime was created as an extension of SGML to be able to describe the structure of multimedia documents.

HyTime extends SGML by adding "architectural forms" which can be used as building blocks to create hypermedia documents. These architectural forms are grouped into six modules:

- The base module contains the architectural form making up the document itself and the activity tracking mechanisms.
- The measurement module provides constructs which allow a document to deal with the concepts of dimension, measurement, and counting. Units of measurement are defined and can be used to specify locations on particular dimensions.
- The location address module provides constructs to specify locations in documents. The module supports, for example, addressing by name, by position, or by semantic construct.
- The hyperlink module allows the creation of links between locations. In contrast to SGML, links to external documents are allowed. Links to non-HyTime documents are also supported.
- The scheduling module describes finite coordinate spaces (fcs), which are

spaces made up of an unlimited number of dimensions. Events can be associated with locations on the axes of the finite coordinate spaces.

- The rendition module defines mappings from one fcs to another. For example, one fcs representing the logical structure of a document can be mapped into another fcs representing the physical structure.

In the HyTime architecture, a hyperdocument is parsed by a SGML parser, whose output is fed to a HyTime engine, which in turn interprets the received data and passes the result back to the application using the HyTime engine.

Several languages have been proposed for querying HyTime documents, e.g. the HyTime query language HyQ [Kimber, 1993] or HyOQL [Gardarin and Yoon, 1994]. One prototypical implementation of the HyTime Standard is HyOctane [Koegel *et al.*, 1993].

The focus of HyTime is on the interchange of hypermedia information. Since this is an application centered point of view, application independent knowledge cannot be represented. The structure of hypermedia data, especially in respect to the content of hypermedia data, is of no interest to HyTime. But since this is one of the focal points of a model aimed at describing the full meaning of hypermedia data as well as its usage, especially the controlled reuse of valuable hypermedia data objects, HyTime is not a suitable data model for a hypermedia database management system.

3.1.3.2 MHEG

MHEG [Meyer-Boudnik and Effelsberg, 1995; Price, 1993] is the international standard for multimedia and hypermedia information objects developed by ISO's Multimedia Hypermedia Expert Group. Its goal is to provide a standardized way to describe complex, interactive multimedia presentations. Other related standards by the same ISO group include JPEG for still images and MPEG for motion pictures (including audio).

MHEG provides:

- Abstractions for real-time presentations including multimedia synchronization and interactivity. Multimedia data is mapped onto a timeline, starts and stops of media objects can be used as triggers for starting or stopping of other media objects.
- Abstractions for real-time interchange with minimal buffering using normal speed data communications.
- Abstractions for direct manipulation of information (called interactions) without any additional processing. Interactions can either be selections, i.e. pushing of buttons, or modifications, where the user types data into an entry field.
- Linking facilities between elements of composite multimedia objects. Links are used to start and stop the playback of media objects, they are triggered either by events raised by other media objects or by selections.

This standard is centered exclusively on modeling presentations, interchange of sets of multimedia objects between applications, and playout management. There are no provisions to model semantic relationships between multimedia data or entities, only temporal relationships can be modeled at a very low level and without any indication of the meaning of the relationship.

3.1.3.3 ODA

The Office Document Architecture (ODA) and Office Document Interchange Format (ODIF) aim to standardize the exchange and processing of electronic documents in the context of office information systems [Bleisinger *et al.*, 1991].

ODA allows the specification of two structures:

- The logical structure, which divides the content of the document into segments based on the meaning attributed to them by the author. This is similar to the structure described by SGML markups.

- The layout structure, which consists of hierarchically nested rectangular blocks. Blocks can contain graphics, images, or text.

Both structures are strictly hierarchical.

Since ODA is intended for office information systems, the standard also defines documents like business letters, reports, and forms.

For the purpose of hypermedia modeling, ODA suffers several disadvantages: The goal of the layout process is to get a set of pages, which cannot be suitably adapted for screen display and which allow only very simple interactions, i.e. page based navigation. Modeling of time based media types is not supported. The logical and layout structure are not completely separated, thus changes in the layout structure can lead to changes in the logical structure. There are no provisions for selective hiding of media segments and segments cannot be reused.

3.1.3.4 Discussion

The data models and standards discussed in this section focus only on *documents* and their representation and thus have a purely application centered point of view. It is impossible to model the *structure* of hypermedia data (in contrast to the structure of the document) and to make it accessible to a wide range of applications. Reuse of hypermedia data, if possible at all, is done in an uncontrolled manner with simple references.

3.2 Hypermedia Databases and Data Models

This section reviews related research in the area of database management systems and data models for hypertext, multimedia, and hypermedia data. Some systems are also described in Section 4.2, which discusses how relevant systems deal with the issue of data independence.

The first section discusses briefly why conventional data models are not suitable for hypermedia modeling. The second section describes survey articles dealing with hypermedia database management systems and data models. The third section contains a detailed discussion of database management systems and data models for hypertext, multimedia, and hypermedia data.

3.2.1 Semantic and Object Oriented Data Modeling

Semantic data modeling, with the Entity Relationship Model (see e.g. [Hull and King, 1987]) as its best known example, has been used successfully in a wide area of application domains. Yet these data models alone are not suitable to model all aspects of hypermedia data. To be able to model the full meaning of hypermedia data, existing data models have been extended in several directions. For example, the HDM model discussed below is based on ER primitives, but adds additional primitives to be able to express navigational semantics. Other common extensions add modeling of temporal relationships between multimedia data objects.

The same is true for object oriented data models and design notations. They too have to be extended, with special class hierarchies and new kinds of relationships, to capture the full meaning of hypermedia data. For example, while object oriented methods are suitable to capture the semantics of multimedia type hierarchies [Lange, 1994], they fail to properly describe the object interactions, which are important in hypermedia applications. For this reason, in the Enhanced Object-Relationship Model (EORM) [Lange, 1994] (for a discussion of

EORM see below) the Object Modeling Technique (OMT) [Rumbaugh *et al.*, 1991] is extended by adding attributes and behavior to relationships to capture inter-object semantics.

3.2.2 Survey Articles

This section gives a brief description of a selection of survey articles dealing with research in the area of database management systems and data models for hypertext, multimedia, and hypermedia data.

- [Narasimhalu, 1996] contains an interesting survey about research issues, challenges, methods, and data models. The article contains a chronology of research in the area of multimedia database management systems and also contains references to other, earlier survey articles and magazine issues dealing with multimedia database management systems. The article also identifies contributions from the field of vision research (pattern recognition), database research, information retrieval, and expert systems. The survey section concludes with an overview about the state of the art of commercial systems.
- [Conklin, 1987] introduces the concept of hypertext and gives a survey of early hypertext systems. Several features of hypertext systems are identified and the systems are compared according to those features. The article also discusses the advantages and disadvantages of hypertext.
- [Boone, 1992] contains a survey of early hypermedia data models, e.g. KMS [Akscyn *et al.*, 1988] or HAM [Campbell and Goodman, 1988]. After a brief description of the data models, modeling characteristics like low level abstractions, high level abstractions, and operations of each model are identified and compared. The article also surveys topics like versioning, collaboration, interoperability, interchange, and security.
- [Woodhead, 1991] surveys the history of (early) hypermedia systems from the very beginning with Bush's Memex [Bush, 1945], the beginning of hy-

pertext [Nelson, 1965], early hypertext implementations, to first implementations of hypermedia systems.

3.2.3 Hypermedia Data Model and DBMS

This section contains a discussion of data models and database management systems which have been created to be able to model and store hypertext, multimedia, or hypermedia data.

3.2.3.1 Dexter Hypertext Reference Model

The Dexter Hypertext Reference Model [Halasz and Schwartz, 1994] is intended to be both: a basis for comparing hypertext systems and a basis for developing interchange and interoperability standards.

According to the Dexter model, hypermedia systems are divided into three layers (from bottom to top): the within-component layer, the storage layer, and the runtime layer.

- Within-component layer: The within-component layer deals with the content and the structure *within* hypermedia nodes. This layer has intentionally not been defined: it is the responsibility of the tools manipulating the data, which make up the within-component layer, to deal with the requests issued by the storage layer.

- Storage layer: The storage layer is the core of the Dexter Hypertext Reference model and describes a network of links and nodes.

  The basic unit of storage is the component, which is a persistent object. A component has a unique identifier and consists of two parts: component content (the actual data) and component information. Component information consists of sets of attributes, a presentation specification, and a set of anchors. Anchors are the interface between the storage and the within-component layer. Anchors consist of an identifier, which is used by link components (see below) and a value, which is to be resolved by the corresponding tool in the within component layer into a "region" of the media data. The definition of "region" depends on the data type of the media data: a region of a text might be a sentence, a region of an audio recording might be a range of samples.

Three subtypes of component exist:

- Atomic Component: Atomic components can be compared to the "node" of node-and-link models.
- Link Component: Link components connect components to other components. Links consist of a list of specifiers: each specifier has a presentation specification, a direction (to, from, bidirect, none), and an identifier for the component and anchor it is attached to.

  Presentation specifications are the interface between the storage and the runtime layer. Presentation specifications depend on the presentation tool, the component being presented, or the access path to the component, i.e. which link has been used to reach the component. Presentation specifications can, for example, consist of screen location, screen size, or presentation mode (read-only, editing allowed) of the component.
- Composite Component: Composite components allow the hierarchical structuring of components.

A hypertext is a network composed of hierarchies of components which are interconnected by links.

A formal definition of the storage layer is given in [Halasz and Schwartz, 1990].

- Runtime Layer: The runtime layer describes the mechanisms for end-user interaction. It supports accessing, viewing, and manipulating of hypermedia networks, thus the runtime layer captures dynamic and interactional aspects.

A series of interactions with a particular hypertext is called a session, an interaction with particular components is called an instantiation.

One extension of the Dexter Reference Model is the DeVise Hypermedia (DHM) System [Grønbæk *et al.*, 1993; Grønbæk *et al.*, 1994]. DHM extends the Dexter Reference Model to be able to deal with the requirements of cooperative work. DHM adds long term transactions, flexible locking, event notification, and version control.

Another extension of the Dexter Reference Model is the Amsterdam Hypermedia Model (AHM) [Grønbæk and Trigg, 1994]. AHM addresses the problem of temporal modeling, which has not been addressed in the Dexter Reference Model. Nodes are grouped into composites. Temporal relationships are modeled by start offsets of nodes from the start time of the composite and by synchronization arcs. Start offsets are used for coarse-grained synchronization, synchronization arches for fine-grained synchronization. For the presentation, this structure is mapped into channels which can then be manipulated by the user, e.g. choosing between alternate channels or VCR-style controls (stop, start, fast forward, ...). A detailed description of the issues related to embedding video in hypermedia documents modeled with AHM is given in [Bulterman, 1995].

3.2.3.2 Hypertext Abstract Machine

In [Campbell and Goodman, 1988] HAM, a transaction-based server for hypertext storage, is described. HAM provides storage and access mechanisms for nodes and links. Nodes can contain any kind of data, the content of a node is not interpreted by HAM. Links cannot be attached to arbitrary regions within a node, but only to a point defined by an offset. The offset is interpreted according to the kind of media data, e.g. for a text the offset is given in characters from the beginning of the text. HAM allows distributed, multiuser access and transactions. Versioning is supported by maintaining a complete version history of the hypergraph and enabling the user to access any version. Unlimited numbers of attribute/value pairs can be attached to links and nodes, these attribute/value pairs can be used to traverse or query the network of links and nodes.

3.2.3.3 Neptune

The Neptune system [Delisle and Schwartz, 1986] is build on top of HAM. Neptune extends HAM by providing an application layer which supports large CAD applications as well as source and object code in a CASE (Computer-Aided Software Engineering) system. The Neptune system implements the following documentation user interface:

- Graph browsers show a (part of a) network of links and nodes graphically.
- Document browsers show the content of nodes and allow navigation operations in hierarchically structured documents.
- Node browsers allow editing of the content of nodes as well as creation or traversal of links.

3.2.3.4 Garg's Set Theoretic Model

Garg [Garg, 1988] presents a set theoretic model of hypertext. The purpose of the model is to demonstrate several useful concepts of abstractions in hypertexts. Nodes in the sense of the Dexter Reference Model are represented by information objects, which are instances of domain objects. Information objects cannot exist without associated domain objects. The information in the object is a set of position/value tuples which are called "lines". Links are unidirectional, their source and target are also modeled as "lines".

This model is used to introduce abstractions like aggregations (a collection of objects which can be referred to by an identifier) and generalizations (a collection of objects which is referred to by a generic object which captures the essential similarities between the objects). Garg also incorporates revisions (versions) and keyword-based filtering.

The model is mathematically well founded. Yet the lack of explicit and flexible modeling of relationships makes its use in the context of hypermedia modeling difficult.

3.2.3.5 Intermedia

Intermedia [Yankelovich *et al.*, 1988] is an application framework which provides hypermedia facilities. Intermedia is an extension to the MacApp application framework. Implemented applications include a text editor, a graphics editor, a scanned image viewer, a three-dimensional object viewer, and a timeline editor.

The MacApp application framework is extended by functionality enabling linking, e.g. creating and traversing links or highlighting anchors. Intermedia also provides "building blocks" for viewing and editing of text, graphics, and tables. Because of the object oriented implementation and the application framework approach, new applications can be developed fast and in isolation from the working system and are then simply linked into the working system.

Intermedia provides span-to-span linking, where each application defines the meaning of "span" for its documents. Intermedia supports bidirectional links. Link types provided include relations, which are primary paths, and references, which are secondary paths—similar to footnotes or "see also" references. Predefined keywords can be associated with links, and can then be used to query the hypermedia web. Links have an "explainer" attribute, which describes the purpose of the link textually. Also supported are annotations by the user. Intermedia is a multiuser system and offers access restrictions and concurrency control.

The Intermedia data model is very simple and uses only plain nodes and links to model information and their relationship. No semantic modeling or temporal modeling is possible, no additional attributes can be defined for nodes or for links.

3.2.3.6 Hyperbase

HyperBase [Wiil, 1993] is a hypertext database management system supporting collaborative work and is based on the client-server model. Another goal was easy extendibility and adaptability for a wide range of applications. HyperBase consists of the following levels:

- The "Basic Entities" level provides the interface to the underlying file system. Hyperbase only supports a very simple link and node model: links are unidirectional and can point to only one node, links pointing to links are not allowed. Each node can be the source for an unlimited number of links. One interesting construct is the many-to-one link, i.e. several nodes can be the source of the same link. The links are node-to-node links, i.e. there are no anchors which allow span-to-span links.

  HyperBase supports RCS [Tichy, 1985] style versioning of nodes. Links and therefore the structure can *not* be versioned.

  Predefined key/value pairs can be associated with links and nodes. One of the key/value pairs is the data associated with the node. The data is treated as a binary stream of data and is not interpreted by HyperBase.

- The "Basic Services" level provides the basic operations on the network of links and nodes:

  - Creation and deletion of links and nodes.
  - Reading and writing (updating) of the content of the node.
  - Creating, reading, writing, and deleting of the attributes of the links and nodes.
  - Version management for the nodes.
  - Management of the connections between HyperBase and the clients.

- The "Multiuser Services" level supports collaborative work by providing an event notification mechanism and a locking mechanism.

The event mechanism notifies users of specified operations being performed on specified (sets of) links and nodes.

The lock mechanism allows the locking of specific key/value pairs, and thus also of the media data, for updates.

HyperBase supports only a very simple data model, offers only node-to-node links and offers no form of temporal modeling.

3.2.3.7 Hyperform

Hyperform [Wiil and Leggett, 1997] offers a very flexible development environment which enables the rapid development and prototyping of hypermedia applications and hypermedia systems. Hyperform is a successor to the HyperBase System discussed above.

The emphasis of the work is placed on creating a collaborative Hyperform development environment which provides the necessary building blocks for advanced hypermedia systems and which can be extended easily. For this reason no specific data model is enforced.

The implementation of Hyperform is based on a Lisp dialect and on custom c libraries.

The Hyperform development environment consists of four types of components:

- Hyperbase management server: the server offers a highly flexible system, where objects and classes can be created or modified at runtime. The server also offers a sophisticated attribute system: attributes can be added or deleted dynamically and objects of the same class can have different additional attributes. Attributes can be read-only, read-write, write-once, calculated, or dynamic.

The server offers concurrency control, notification control, access control, version control, and query and search facilities.

- Tool integrator: the tool integrator provides an extensible, tailorable interface between the server and the participating tools. The tool integrator can be extended during runtime and thus allows the integration of new participating tools while running. The tool integrator also provides a cache for the participating tools.

- Editors: The editors are used to create and modify class definitions and thus can be used to customize the provided database management services. Currently GNU Emacs [Stallman, 1995] and the EHTS Editor [Wiil, 1991], which is based on an extension of GNU Emacs, can be used. With these editors system developers can collaborate in the design process of a new hypermedia database management system.

- Participating tools: no specific examples of participating tools are mentioned. Hyperform can cope with internal tools, which are loaded into the hyperbase management system, and external tools, which communicate via Unix and network libraries with the tool integrator.

As a proof of concept, Hyperform has been used to implement the Dexter Hypertext Reference Model [Halasz and Schwartz, 1994].

Hyperform offers a number of useful and innovative services and allows the development of systems in a collaborative manner. Yet it is not clear why especially hypermedia applications should benefit from Hyperform. Hyperform offers no specific functionality for hypermedia: neither playout nor media storage for continuous retrieval (see Section 3.1.1) are provided.

The decision not to support a (specific) data model has several unpleasant consequences because Hyperform cannot support capabilities which are specific to hypermedia systems:

- Hyperform offers RCS [Tichy, 1985] style versioning of *data.* No versioning of the structure, i.e. of the links, is provided, although structure versioning (see for example the results described [Durand *et al.*, 1994]) is a valuable service for hypermedia database management systems.

- Hyperform offers unix style access control (access rights are defined for owner, group, or all users), but again only on the node level. Any higher form of access control, e.g. access to whole subnetworks or access according to certain semantic properties, has to be implemented by the user of Hyperform.

- Hyperform offers basic query support like general filtering and list based search as well as access to meta data. Hypermedia specific queries are not supported.

3.2.3.8 MODEL

The multimedia-object data model (MODEL) is a layered data model for multimedia objects proposed in [Narasimhalu, 1996]. The data types level describes media types such as images, music, speech, video, as well as general types such as strings and integers. The object types level contains objects like audio, video, bag, collection, temporal, and spatial. Object types can have default attributes, which can be modified by the user at the time of object creation. The topmost level describes relationships between objects. Examples of relationships include spatial, temporal, or transcoding (e.g. speech-to-text) relationships. Relationships have a type (for example spatial) and a value (for example above, below, left, right, ...). The article also introduces a graphical notation for the data model.

The data model is intended to model technical aspects of multimedia data, e.g. resolution, or how multimedia data is composed. It is not intended to model the content of multimedia data. The article does not give an indication what information is modeled in the schema of the database management system, and what information is modeled at the object label. According to the examples, the schema will only contain generic object types for various kinds of media.

The article also describes a transaction model, a query model, a storage model, and an interface model, each model model consisting of several managers. The whole architecture contains five more managers, an integrity rule base, and an undefined but large number of cooperating agents and submanagers.

There is no indication of any plans to implement this architecture, only a vague hope that the architecture is a good starting point and can be refined.

The sheer size and complexity of the architecture is impressive. Unfortunately there is no indication if a complete specification of the architecture and of the data model exists or if a complete version is planned at all. Since neither a complete version nor a prototype of this architecture exist it is difficult to judge the overall suitability of the approach.

3.2.3.9 ABC distributed hypermedia collaboration

[Shackelford *et al.*, 1993] describes distributed graph storage (DGS), which is a part of the ABC distributed hypermedia collaboration environment. DGS consists of four levels (from top to bottom):

- The applications using the system, e.g. editors and browser.
- The application programming interface which decouples the applications from the hypermedia system.
- The graph cache manager, which handles the representation of the links and nodes.
- The storage level, which deals with the persistent, distributed storage of the multimedia data.

The data model models the data only on the instance level and consists of links, nodes, and composites. Links and nodes are used in the usual way to create networks. Composites contain subgraphs of these networks. Each link and each node belongs to at least one subgraph. Subgraphs can be strongly typed to conform to the shape of trees or lists. Both links and nodes can have attributes. Attributes can either be common attributes, which are generally visible, or graph attributes, which are only visible in the context of a specific subgraph. There are two kinds of links: s-links which model the essential structure of a document, and hs-links which model relationships among basic structures. In the DGS model not only nodes, but also links can be associated with media content. The article does not give any indication how the content of a link is intended to be used.

The emphasis of the DGS system is on collaborative work, thus DGS provides access control and concurrency control.

Neither nodes nor links can subclassed, so the modeling of information beyond a simple node and link model is not possible. The problems of hypermedia modelling, especially temporal modeling, are not addressed.

3.2.3.10 HyperBase3

HB3 [Leggett and Schnase, 1994] and its predecessor HB1 [Schnase *et al.*, 1993] provide a framework for the distribution of hypermedia across wide area networks and heterogeneous platforms based on a set of cooperating processes.

The SP3 hypermedia model implements the idea of "hypermedia-in-the-large", a coupled set of processes which together provide a flexible, distributed, extensible hypermedia system. The hypermedia model consists of the following building blocks:

- Applications are programs displaying or modifying hypermedia data.
- Components contain the information managed by the applications. Components correspond to nodes in the terminology of the Dexter reference model.

- Persistent selections denote certain parts within a component and thus correspond to anchors in the terminology of the Dexter reference model. Persistent selections are managed by applications.
- Anchors are *processes* which are associated with persistent selections.
- Links are *processes* which are associated with anchors.
- Associations are objects used to model the relationships between anchors and persistent selections and the relationships between anchors and links. Associations are grouped into association sets to provide multiple contexts.

The architecture consists of the following parts:

- Participating applications are used to view, edit, browse, and manipulate hypermedia data. Participating applications have to provide a set of call-back routines to implement hypermedia functionality. The HB3 system provides an interface to a high-level, policy-neutral link service toolkit.
- The link services manager provides runtime support for interapplication linking. Communication with the participating applications is maintained via IPC (inter process communication). The link service manager supports the attachment and detachment of links, requests applications to display anchor and link markers, and supports browsing operations and context operations.
- The association set manager is responsible for providing persistent and shareable storage for associations and association sets.
- The versioned object manager provides version histories for all objects which are stored in the hyperbase.
- The storage manager maps the HB3 hypermodel into physical storage.

The HB3 hyperbase management system consists of the association set manager, the versioned object manager, and the storage manager.

The SP3 model uses semantic modeling [Hull and King, 1987] to model the storage of hypermedia data. Neither the content of the data nor the meaning of the relationships between the nodes are modeled. There are no provisions for temporal modeling.

3.2.3.11 HyperStorM

HyperStorM [Bapat *et al.*, 1996] is an extensible object oriented hypermedia engine. In this context, a hypermedia engine is an abstract machine, which provides an interface for accessing and manipulating hypermedia structures. HyperStorM is an extension of the OODBMS VODAK.

HyperStorM uses meta classes to provide hypermedia modeling capabilities, e.g. modeling of link-relationships, whole-part relationships, inheritance relationships, and role relationships. The meta classes are used to instantiate application classes, which model hypermedia data for a single application. Instances of application classes contain hypermedia data.

There are no provisions to model temporal relationships.

HyperStorM divides the data in a hyperstructure level, which models a network of links and nodes, and in a content level, which contains multimedia data. The presentation of the hypermedia data is outside of the scope of HyperStorM, there are no provisions to achieve physical or logical data independence.

3.2.3.12 GRNL model

The GRNL model (grain/role/node/link model) [Arnold *et al.*, 1994b; Arnold *et al.*, 1994a] is intended to facilitate easy administration of semantic connections, which the authors call the “hyper” aspect of hypermedia.

The data model consists of the following parts:

Nodes: nodes are information chunks and contain the data to be displayed. A typing mechanism, which is not specified in the paper, "can be used to reflect the nature (both media and semantic) of nodes involved in connections."

Grains: are locations within a node which are anchors of a semantic connection. The difference between "anchor" and grain is explained below.

Grains can be typed and nodes of a certain type can be restricted so that only grains of a certain types can be used. For example, certain texts are only allowed to have grains which are words or whole paragraphs, but not arbitrary sequences of characters. The type system for grains is not specified.

Roles: Grains fulfill a specific role in the context of a link. Roles can be identified by labels, which should be semantically meaningful component parts of a link.

Links: Links are labeled structures, where again the label should reflect the relationship between the grains. Links are collections of roles.

The GRNL model is implemented in an object-oriented way using Smalltalk and Gemstone, yet the only classes which can be subclassed are the "node" classes. Thus no additional information can be added to e.g. links. Grains and roles are not implemented as objects but as attributes, because they cannot exist without the node or link they belong to.

All nodes are typed in two ways: (a) by the media data they contain and (b) by the semantics of their use. The type of a node is used to define constraints of the relationships between nodes, e.g. how many relationship occurrences are allowed with a node of a certain type and if relationship occurrences with nodes of a certain type are allowed at all. Types are only a collection of templates for links, specifying the constraints mentioned above. Neither attributes of types are modeled, nor the content or the structure of the media content of a node are modeled.

For the implementation two new concepts are introduced:

Anchors: Anchors are "wrappers" which hold together a grain, which is an attribute of the anchor, and the node the grain belongs to.

Link designs: contains specifications of links which are legal between certain types of nodes.

Many of the design decisions are not explained, especially why only nodes can be subclassed and why grains are not simply related to the node with a part-of relationship. Then the concept of a "grain" could be dropped in favor of the familiar concept "anchor".

One other shortcoming of the model as noted by the authors is that "It does not provide a way of creating overviews of the information within a hypermedia document." A much more severe shortcoming is that the model fails to "focus on the multimedia nature of hypermedia systems"—there are no provisions for temporal or spatial modeling.

The GRNL model is also used for the specification of local graph-based constraints in hypermedia systems [Arnold *et al.*, 1994b]. The LHGB model (labeled hypergraphs with blocks) is an extension of the GRNL model. The GRNL model is extended with blocks and behavior:

Blocks are labeled set of nodes, where the label is the same as the label of roles.

Links are labeled sets of blocks.

In this paper "blocks" seem to have taken over the role of "roles" as described above.

Upon this model an algebra is defined, and the algebra is then used to define constraints. Also algorithms for the detection of violations of the constraints are described.

The authors claim that this approach can help people to develop an overall understanding of the structure of a big hypermedia web by reducing the complexity of the networks. Yet all the approach does is to control a given level of complexity, no procedures or rules to create more manageable hypermedia webs are given. The authors also mention that one important task of the future work is to find "a compelling example of a hypermedia system which would be improved by the use of the above defined algebra and constraints.", so it is not clear if the approach can be applied beneficially.

3.2.3.13 Enhanced Object-Relationship Model

A formal model for hypertext is presented in [Lange, 1990], where the Vienna Development Method (VDM) is used to model hypertext. This model deals with textual data *only*.

A hypertext consists of a network of links and nodes. Nodes can be composed of "slots" which are named regions of text. The model supports both node-to-node, slot-to-slot, and span-to-span links. Links are directed, but can also be browsed in the "reverse" direction. The model distinguishes between source and destination anchors, yet it is unclear why the distinction is made. N-ary links connect a set of sources to an (ordered) set of destinations and can be used to model sequences. Attribute/value pairs can be attached to all constructs of the model.

This model also supports a construct called "network", which is a subset of the set of all links and thus allows to model a subset of the whole network of links and nodes. This construct can be considered to be a simple view mechanism. Existing nodes and networks can be restructured into tree-structures, set-structures, or map-structures, which are intended to provide easier access to the user of the network. This again can be considered as a simple view mechanism.

[Lange, 1990] also describes an object oriented class model which implements the data model.

The model is restricted to textual data and offers no support for temporal modeling.

In an extension to the work presented in [Lange, 1990], the Enhanced Object-Relationship Model (EORM) [Lange, 1994] extends OMT by adding attributes and behavior to relationships to capture inter-object semantics. Examples for inter-object semantics include chapter number attributes, preview mechanisms for citation relations, access control, GUI related semantics (presentation style), trigger functions, or constraint checking. Here, the concept of a node is not used. Objects, which are instances of classes, are related by links.

EORM is used to model the results of a (general) object oriented design method, which can also be applied to hypermedia applications. The design method consists of the steps of identifying classes, refining classes, adding operations, and introducing inheritance.

The model is intended for modeling hypermedia application systems, not for the modeling of the content and the structure of hypermedia data. The approach deals only with predefined elements: strings, text, numbers, buttons, bitmaps, and presentations. Presentations consist of collections of fields which are used to display elements. Thus, at least by the definition given in Section 2.1.3, this approach does not provide sufficient capabilities to cope with hypermedia data. While it provides a foundation for more expressive object oriented modeling, it fails to directly support hypermedia specific issues, e.g. temporal modeling.

3.2.3.14 Trellis Model

The Trellis model of hypertext [Stotts and Furuta, 1989] is based on Petri nets, where nodes are modeled as "places" and links are modeled as "transitions". "Tokens" mark the "places" which are displayed to the user. This approach allows to exploit the algorithms developed for Petri nets, e.g. algorithms for the detection of node unreachability, and the application of concurrency as modeled in Petri nets to hypertext modeling.

In addition to places and transitions the Trellis model uses sets of "contents", "windows", and "buttons", which are human-consumable, and two sets of mappings, called "projections", between the Petri net, the human-consumable objects, and the display mechanism.

The Petri net models only the network, the content of the node, the "content" mentioned above, is not considered. A logical projection maps "content" to "places", "places" to "windows", and "transitions" to "buttons". A display projection maps "buttons" and "windows" to display devices. The display projection also determines where the button is displayed on the display device. This means that this projection has to know which span to text is associated with a button. Since this knowledge is clearly structural it should be stored on the structural level of the Petri net.

"Places" can contain whole hypertexts, which allows the modeling of hierarchies.

Using certain patterns of "places" and "transitions", access control and multiple versions of the same hypertext can be modeled. Yet no higher abstractions are provided or can be created so the designer has to rely on external documentation to be able to understand the meaning of a particular pattern. Thus the expressiveness of the Trellis model is low. A proof of concept prototype called αTrellis was implemented on Sun Workstations.

The Trellis hypertext reference model (r-model) [Furuta and Stotts, 1990] is a meta-model for the Trellis model described above. The r-model distinguishes between an abstract hypertext, a concrete hypertext, and a visible hypertext.

- The abstract hypertext specifies a network of links and nodes and the content of the nodes, but does not describe how the hypertext will be displayed. The abstract component level describes structure, content, buttons (anchors), and containers (a description *how* links and nodes are aggregated, e.g. if they are displayed simultaneously or combined into one window). The abstract hypertext level provides associations between content and structure, buttons and structure, and container and structure.

- The concrete hypertext provides a mapping between components of the abstract hypertext and (logical) windows. This level also describes how buttons are displayed.

- The visible hypertext maps the concrete hypertext to an unlimited number of (distributed) displays and users.

Later the Trellis model was extended with colored tokens and transition timing [Furuta and Stotts, 1994a; Furuta and Stotts, 1994b]. Tokens now have a color (type) and may also have a data structure. Colored tokens make it possible to distinguish between different participants or groups of participants who concurrently use the hypertext. For each transition two time values can be specified: the release time specifies the number of time units which must pass before the transaction is enabled, the maximum latency specifies the maximum number of time units, starting when the transition has been enabled, after which the transition fires automatically.

These extensions are incorporated into a new prototype called χTrellis which is used to demonstrate that formal requirements specifications from software engineering as well as process simulation can be modeled, visualized, prototyped, executed, and validated with the Trellis model. χTrellis is implemented using a distributed client/server architecture, where the server is an engine managing the Petri net. The server communicates with various clients using remote procedure call (RPC) requests. For the combination of the modeling of the structure and the content (the "user-manipulatable information") and the modeling of the process (the "user-directed execution") the term "hyperprogram" is introduced.

3.2.3.15 Transient Hypergraph Model

A data model based on transient hypergraphs is presented in [Watters and Shepherd, 1990]. Hypergraphs are created on demand, their structure being defined by a query. Hypergraphs are virtual structures which are built out of dynamically instantiated links between existing nodes. Hypergraphs exist for the duration of a query or a query session.

Entities in the database are represented as nodes of the hypergraph. Those nodes, whose attributes satisfy a given condition, are grouped into sets. These sets describe relationships between nodes and are represented as hyperedges in the hypergraph. For browsing purposes the hyperedges can be linearized, so that each item which is an element of a hyperedge is connected to each other item of the hyperedge through a link. Set operations like intersection, union, and difference can be used to create new hyperedges from existing hyperedges.

The advantage of hypergraphs is their ability to allow the user to browse through the results of a query. Hypergraphs can be used on top of any database or information retrieval system.

The presented model deals only with instances of a single entity, the implications of allowing instances of multiple entities in queries are not discussed. Another disadvantage is that the model only supports node-to-node links.

In [Watters and Shepherd, 1990] hypergraphs are not used to model the schema of the data, but to provide a model of data access. The information which is stored in the schema of a database cannot be modeled with this approach.

3.2.3.16 Hypergraph Data Model

In [Tompa, 1989] another model based on hypergraphs is presented. In contrast to the approach presented in [Watters and Shepherd, 1990] this data model describes the schema of a hypertext database.

One goal of the model is to achieve a decoupling of content and structure, for this reason a function is used to map nodes, which are part of the structure, to pages, which contain the content. A page is considered to be atomic, its internal structure is not considered in the context of the data model. Thus only node-to-node linking is possible. A function maps pages to applications which can display and manipulate the data contained in the pages. The hyperedges are composed of a set of source nodes, a set of destination nodes, and a set of labels. A label can be any atomic value which can be tested for equality, thus also audio or video segments can be used as labels.

The article also discusses the implementation of the data model using a relational database management system and the methods necessary to create and manipulate hyperedges.

Even if the author claims to be able to model all kinds of hypermedia data, the problem of temporal modeling is not addressed at all.

3.2.3.17 Hypertext Design Model and Successors

The Hypertext Design Model HDM [Garzotto and Paolini, 1991; Garzotto *et al.*, 1993d] and its successor HDM2 [Garzotto *et al.*, 1993b; Garzotto *et al.*, 1993a] are used as the design notation of a model based approach to hypertext application design.

HDM applications consist of structures of information chunks called "entities". Entities are grouped into "entity types". Entities are the smallest *independent* pieces of information in the context of HDM. An entity is a hierarchy of "components", which in turn are made of "units". Each unit belonging to a certain component contains the same information, but has a different "perspective", e.g. one unit contains a german text and another unit contains a translation of the german text into english, or one unit contains an image and another unit contains the corresponding textual description. Which perspectives and which incoming and outgoing links are available is described at level of the entity types.

The authors state that HDM is *not* indended to be a semantic data model, thus it does, for example, not distinguish between hierarchies of entities with different meaning.

Inheritance relationships between entity types are not discussed, thus object oriented modeling is not possible. Due to the complex structure of the entities it is not clear how inheritance can be used.

HDM defines the following types of links:

Structure links: connect components within the same entity.

Perspective links: connect different perspectives (units) of the same component.

Application links: connect components of different entities.

Various constructs are used to adapt a hyperbase to the needs of its users:

- Outlines: Outlines are hierarchically ordered trees of components, where only the leaf nodes are allowed to have links to entities or components in the hyperbase. Outlines are not considered to be part of the hyperbase schema.
- Derived links: derived links can be computed from the structural relationship of the components (derived structural link types) or from application links using operations like "inversion" or "composition". This aspect of the model is only coarsely sketched and not fully specified. Since HDM is no semantic data model and since the derivation is not very powerful and has to be specified manually, this approach is weaker than the propagation of relationships as proposed in Section 5.4.3.4.

Another aspect of HDM are the "browsing semantics" of the hypertext, which define which information objects are used for "human consumption", how the links between the objects are displayed, and which behavior is activated by those links. Basically, the HDM approach provides a mapping from the HDM schema to (plain) nodes and links which can be implemented in any hypermedia platform—a browsing semantic is an interpretation of a HDM model in terms of links and nodes.

HDM is intended to model the application domain. The structure of the multimedia data and the relationship between multimedia data objects is not modeled, there are no provisions for temporal modeling. Also, components are grouped in an *application dependent* hierarchy, thus issues regarding data independence are not addressed.

HDM is intended for rigidly structured domains, arbitrary nodes are not allowed and cannot be modeled.

Sharing of units between entities is allowed, but not encouraged, because it is regarded as a possible cause of disorientation of the user. Even if this may be true at the application level, this is certainly not true at the database level, where a mechanism for the controlled reuse of multimedia data is an important goal.

HDM2 extends HDM in several ways:

- The concept of a "link" is generalized into the notion of a n-ary web: webs connect a set of "destinations" (elements like entities or components). Webs can describe the relationships with information stored in an optional center component.

- HDM2 provides access mechanisms which allow the user to conveniently access the data stored in the hyperbase. Access mechanisms have the same structure as the webs described above, but have a fixed meaning. "Indexes" are entry points which allow users direct access to certain objects in the hyperbase and can be compared to a "table of contents". "Guided tours" specify an ordered sequence (a linear path) across a set of objects, which allows the user to navigate in a controlled manner using "next" and "previous" operations.

- In addition to links, also entities and webs (and thus indexes and guided tours) can be derived, the results of the derivations can be collected in "hyperviews".

- The browsing semantic is adapted to the new constructs and described in detail in [Garzotto *et al.*, 1993a]. The authors claim that the different kinds of browsing operations described in this article reduce the cognitive burden of the user of the hypertext by differentiating between different types of links, e.g. structural links (moving up, down, or sideways in a component) or application links.

Hypertext application development consists of two tasks: "authoring in the large" designs information elements and navigational structures, "authoring in the small" instantiates the network of links and nodes and fills in the content. HDM deals with authoring in the large *only.*

A design methodology based on HDM is presented in [Garzotto *et al.*, 1993c]. The methodology consists of the following steps:

1. Hyperbase schema design: defines the entity types and their relationships.

2. Instances design: designs the internal structure of the entity types.

3. Reading schema design: the hyperbase schema is filtered or redesigned for particular applications or groups of users.

4. Access structures design: designs the access structures, i.e. how indices and guided tours are used.

5. Authoring in the small: creating or locating the data to be presented and assigning it to the correct place in the entities.

6. Navigation support design: defines navigation mechanisms, e.g. session histories or the exact runtime semantics of guided tours.

7. Additional gadgets: Examples mentioned in [Garzotto *et al.*, 1993c] include printing facilities or integration with external applications.

8. Screen design: specifies the aesthetic appearance of the application.

One extension of HDM is HDM+ [Cavallaro *et al.*, 1993], which adds authoring in the small in the context of navigational access to information stored in databases. HIFI (Hypertext Interface For Information systems), which uses HDM+, provides a model for a navigational interface to databases, a strategy to map the interface model to the data structures in the database, as well as the architecture of the execution tools. In addition to structural links, access links, and perspective links, HDM+ defines applicative links, which allow the user to move from one entity to another.

The Relationship Management Design methodology (RMD) [Balasubramanian *et al.*, 1994] uses the RMD model to design and model hypertext applications. The RMD model extends the HDM and HDM2 data model by adding conditional indices (only members of a set satisfying a specified condition are presented to the user), conditional guided tours, conditional indexed guided tours, and an additional grouping mechanism for the added structures.

The RMD methodology consists of the following steps:

- Designing the application domain using E-R design.

- Designing the information content of the entities of the E-R design.

- Navigation structures are added to the E-R design and combined with the result of the entities design resulting in a RMD model.

- Designing the conversion process from the RMD model into the application model. This allows the RMD methodology to be used with any application which supports a plain node and link model. For the prototype application described in the article this conversion is done manually.

- Designing the user interface.

- Designing the runtime behavior, for example specifying the operations (e.g. replacing the displayed content or opening a new window) to perform when links are traversed.

- Construction and testing.

The RMD methodology uses the same steps that are used in other design methodologies, compare for example [Lange, 1994] or [Rumbaugh *et al.*, 1991]. The conversion approach allows nearly any available hypertext or hypermedia system to be used to implement the design. Yet the approach is only suitable for applications where the structure of the application does not change: since the content of the nodes is added manually (this task is outside of the scope of the methodology) either all nodes have to be updated manually if the structure of the links is changed, or the structure can be recreated from the design and the content has to be added again. Thus the maintenance of the *structure* of the application is not supported well.

OOHDM [Schwabe *et al.*, 1996] is another extension of HDM which is used in the context of a design methodology. The methodology consists of the following steps:

- Conceptual design: OMT [Rumbaugh *et al.*, 1991] is used to design a model of the application domain.

- Navigational design: Defines a view based on the conceptual design. This step produces two kinds of schemata: navigational class schemata and navigational contexts. The navigational class schema defines which entities of the conceptual schema are visible to the user and how they are mapped into pre-defined navigational classes like nodes, links, or access structures (see also Section 4.2). Nodes can contain selected pieces of data from a set of entities. Navigation contexts are derived from nodes, links, indices, and guided tours and are used to structure the navigation space. Basically, contexts can be considered to be collections.

- Abstract interface design: defines how the various objects will be made visible to the user, which navigational objects will appear, and what happens when those navigational objects are activated. This step provides a template for the display of the hypertext objects. Various ways for specifying these templates are discussed.
- Implementation: Again, the design is mapped to an available hypermedia platform like Hypercard, Toolbook, or even HTML.

All approaches based on and derived from HDM share the notion that first the (navigational) structure is defined, then at a later stage, often outside the scope of the work presented in the articles, the multimedia data is added. This approach is not suitable to store large amounts of hypermedia data and to store its properties, since modeling the structure, both the technical structure (composition and temporal relationship) and the structure of the content of (existing) data is not supported.

3.2.3.18 Hypermedia Data Model

The hypermedia data model (HM data model) is described in [Maurer *et al.*, 1993b; Maurer *et al.*, 1993a; Maurer *et al.*, 1994]. The core of the HM data model are "S-collections" (structured-collections). Each S-collection has a name (a unique identifier), and an optional data unit which contains the media data to be presented to the user. A S-collection also encapsulates an internal structure, consisting of a set of other S-collections. A S-collection has one member of its set defined as the head S-collection which is activated by default.

Links can only be defined between members of the same S-collection. The reason for this restriction is the goal of the authors to provide a hypertext in which the user can navigate more easily, because "We believe that this kind of 'context-dependent' references supports 'coherence' and 'homogeneity' of hypermedia information *(citations omitted)* by browsing." [Maurer *et al.*, 1993a]. Since there are no global links, link maintenance is made easier. S-collections can impose further restrictions on the way their members can be linked together, the following topologies are proposed in the articles cited above:

Envelope: Every S-collection is connected to every other S-collection.

Folder: A sequence of S-collections, each member has a link to the next and to the previous S-collection.

Menu: The head S-collection is connected to every other S-collection and every other S-collection is connected to the head S-collection. This topology is also referred to as "map", especially when the link anchors in the head S-collection are defined by sensitive areas on the screen (e.g. clickable maps).

Freelinks: Arbitrary links between members can be created, but links to S-collections, which are not members of the freelink S-collection, are still forbidden.

Void: contains no other S-collection, and thus also no head S-collection. A void S-collection contains only its content. In contrast to the "complex" S-collections described above, void S-collections are also called "simple".

This is the classification given in [Maurer *et al.*, 1994]. In earlier articles a S-collection does not have a content, and the void S-collection is not mentioned. Instead, the media data is stored in the head S-collection, which then contains a "primitive node".

A S-collection can be a member of any number of other S-collections, recursive member-relationships are also allowed.

Four operations are defined for navigation:

Access: "Executes" one member of the current S-collection by displaying its content to the user. This is the equivalent of following a link in the node and link model.

Zoom_In: Makes the last S-collection which has been "accessed" the current S-collection, displays its head S-collection and makes the internal structure of the new S-collection available for navigation.

Zoom_Up: Allows the user to make one S-collection the current S-collection which *contains* the current S-collection as its member.

Zoom_Back: Is the inverse operation of Zoom_In and Zoom_Up, i.e. it makes the previous S-collection the current S-collection again.

HM also defines operations for modifying the database:

- S-collections can be CREATEd and DELETEd.
- INSERT_MEMBER and REMOVE_MEMBER add and remove members from S-collections.
- INSERT_LINK and REMOVE_LINK add and remove links from freelink S-collections.

To satisfy short-term needs, the navigational structure of a member S-collection can be made available together with the navigational structure of the current S-collection (inclusion), or the navigational structures of two S-collections sharing one member can be made available at the same time (join). The divide operator is the inverse operator to join or inclusion.

Queries can also be used in the context of the HM data model, all S-collections satisfying the given query are entered into a "dynamic" S-collection.

Meta information can be modeled using categories, which are connected with semantic relationships, in [Maurer *et al.*, 1993b] "is a" and "is part of" relationships are mentioned. S-collections can be assigned to any number of categories. The categories and their relationships together are called the "semantic dictionary" and can be used in queries.

In [Andrews *et al.*, 1993] the HM data model is extended with purpose-oriented behavioral abstractions. "Categories" can have several "slots", which only accept S-collections belonging to a certain category. Yet, slots are no structural entities, but define which S-collections can be entered by automatically providing suitable insert and remove operations. The implications of the approach on the internal structure are not discussed. This is strange, since the user friendly representation of the internal structure is one of the focal points of the HM data model. Inheritance between categories (and most importantly, polymorphism) is not discussed. One instance of a category corresponds to exactly one instance of a S-collection. It is not clear why the concept of "category" and "S-collection" are different, since the only relationship which is possible between their instances is one to one. Several constraints for slots are discussed, e.g. unique (only one instance is allowed for the slot), essential (there has to be at least one instance at any time), or compulsory membership (each instance of a category has to be connected to a slot).

The HM data model provides an interesting approach to the modeling of information related by links and nodes. Again, this is a data model aimed at supporting *application* modeling. Temporal modeling is not supported at all, thus the data model does not take the modeling of multimedia data into account. Other topologies of relationships between members of a S-collection are not discussed in the articles, it seems to be tedious to have to define a new kind of S-collection for every link configuration one wants to use.

3.3 Commercial Database Management Systems

This section describes some well known commercially available database management systems with respect to their ability to store multimedia and hypermedia data. Commercial hypermedia authoring systems are not described, because they implement functionality found in clients of a database management system. This functionality is outside of the scope of this thesis. Hypermedia authoring systems often use simple data models which are surpassed by the models presented above.

This section describes the multimedia capabilities of Oracle and Objectstore. Illustra, which is used as the basis of the implementation of a proof-of-concept prototype, is described in Section 8.1. The database management systems have been selected because they are typical examples of state-of-the-art extended relational (Oracle) and object oriented (ObjectStore) database management systems.

3.3.1 Oracle

Oracle is an extended relational database management system. While the basic database management system server offers no multimedia capabilities, Oracle can be extended by several cartridges [Oracle Corporation, 1997]:

- Oracle 8 Image Data Cartridge: This cartridge allows images to be stored as attributes in relations. The images can either reside inside the database or in external files (read only). The cartridge allows the storage in the original image format and offers transcoding of images (format conversion) on demand for a wide variety of image formats. The stored images can also be manipulated. Available manipulations include scaling of images, cropping ("cutting out" of certain areas), as well as manipulation of the compression quality.

- Oracle 8 VIR Cartridge: This cartridge extends the capabilities of the Image Data Cartridge by content based retrieval of images based on image similarity.

- Oracle 8 Video Cartridge: This cartridge offers a streaming video server, which can deliver fullmotion, fullscreen video (mpeg) to multiple clients. Clients can control the playback of videos using familiar VCR controls. The video server has to run on a different server than the database management system server. From the description given in [Oracle Corporation, 1997] it is unclear how the integration of the video data in the video server with the data in the database management system server is achieved.

- Oracle 8 ConText Cartridge Release 2.0: This cartridge allows the storage of text in the database. Supported formats include ASCII, HTML, MS Word, Word Perfect, Adobe Acrobat, Excel, and Powerpoint. Text is stored in attributes. One text can be stored in several parts, where each part is stored in one attribute (called a "detail") and the whole text is represented by one "master"-attribute pointing to the parts. SQL is extended with constructs allowing full text retrieval of stored text. The cartridge supports a number of widely used languages. For english texts theme indexing and theme summaries of the stored texts can be provided.

As described above, the multimedia capabilities of Oracle allow to store certain media types in the database (or in files outside of the database) and also allow a limited number of operations on the stored data. Yet there are no provisions to store the relationships between media data and so there is no specific support for hypermedia data.

3.3.2 ObjectStore

ObjectStore is an object oriented database management system, which can be extended by Object Managers [Object Design, Inc., 1997]. Multimedia data is stored as objects of predefined classes supplied by the object managers, these classes can also be subclassed. Object managers are available for:

- Images: Images can be stored in a variety of formats, conversion into an internal format is not necessary. The image object manager also allows content based retrieval using technology from the same company (Virage Inc.) that also provided technology for the Oracle 8 VIR Cartridge.
- Text: Text is stored in its native format. The object manager provides functionality for content based retrieval and concept based search.
- Audio: Audio data can be stored and played back. Content based retrieval can be done in conjunction with text data which contains information about

the audio data.

- Video: Video can be stored and retrieved. The object manager offers similar capabilities as the audio object manager.

In ObjectStore, all data is stored inside the database. Again, no support for hypermedia is provided.

Chapter 4

Data Independence

As the amount and complexity of hypermedia data grows, a number of problems, e.g. difficulties in the reuse of existing data in different applications, arise from the close connection between the application and the storage data model [Campbell and Chung, 1995]. These problems are comparable to the problems of file based storage and application based data modeling which led to the design of database systems. The advantages provided by database systems are, for example, logical and physical data independence, access based on a common data model, security, backup and recovery [Elmasri and Navathe, 1994].

One of the most important concepts of database systems is data independence. Data independence allows the representation of data to be changed without causing the need to change application programs using the data. Other benefits include a higher degree of security as the user only sees that part of the database he is authorized to access. While the concept of data independence has not yet been fully exploited in hypermedia database management systems, we believe that it can be applied beneficially to achieve a clear separation between hypermedia data and applications.

Thus, one of the main goals when creating the **Hyper**media **D**atabase Management System Architecture was to achieve data independence for hypermedia data.

In this chapter we first describe how data independence is incorporated in conventional database management systems. Then we discuss how physical and logical data independence is achieved in existing hypermedia database management systems.

4.1 Data Independence in Conventional DBMS

This section reviews the concepts associated with data independence by describing the architecture of a typical relational database management system and then distinguishing between logical and physical data independence.

4.1.1 ANSI/X3/SPARC DBMS Architecture

In [Tsichritzis and Klug, 1978] the ANSI/X3/SPARC Study Group on Database Systems describes a generalized framework for the description of database management systems. The framework consists of 18 roles of persons, 23 processes and 42 interfaces. The best known part of the architecture (for an overview see e.g. [Elmasri and Navathe, 1994] or [Date, 1990]) describes the organization of data in three levels, each consisting of one or more schemata (see Figure 4.1):

Conceptual schema: The conceptual schema (sometimes also called logical schema) describes the structure of the whole database.

The conceptual schema uses entity sets, data types, relationship sets, user operations, constraints, security specifications, and integrity checks to describe the structure of the data, but abstracts from the requirements of individual applications and from the restrictions of physical storage devices.

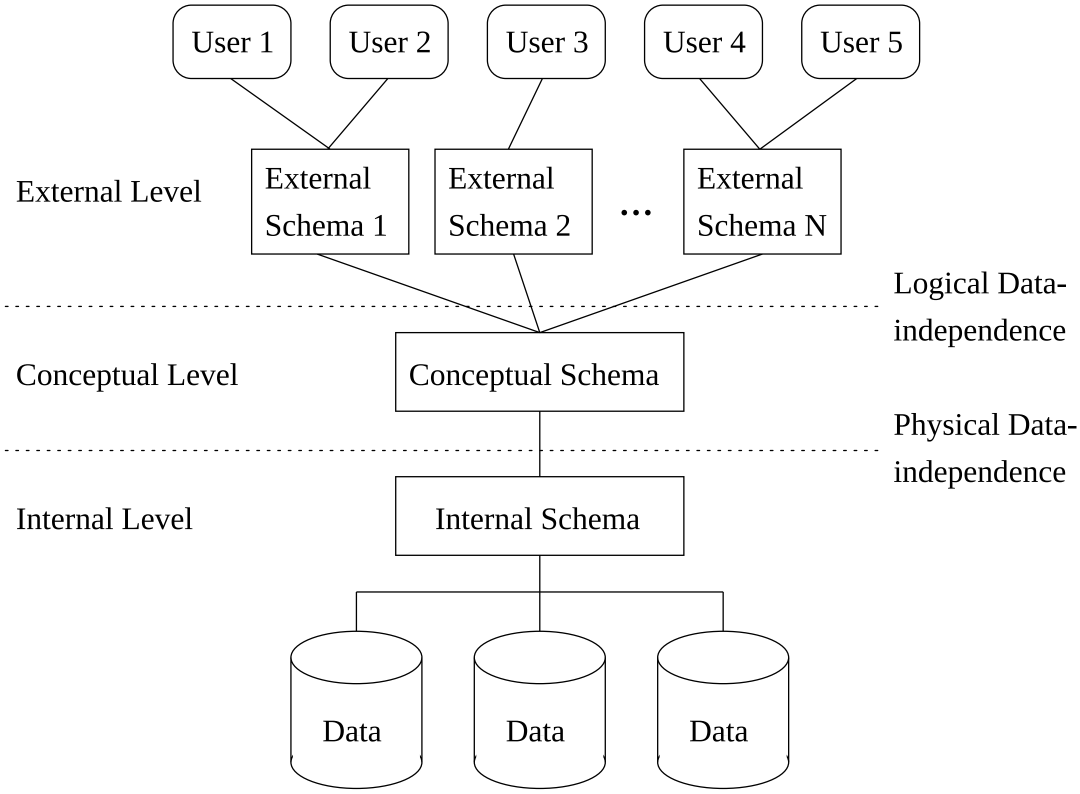

Figure 4.1: Extract of the ANSI/X3/SPARC Architecture

Internal schema: The internal schema describes the physical characteristics of the logical data structures in the conceptual schema.

These characteristics comprise for example the placement of data on physical storage devices, placement and types of indexes, physical representation of relationships between data, and access paths.

External schema(s): An external schema is the view an user or a group of users has of a particular database and is thus a subset of the conceptual schema.

As most users do not require access to all data described in the conceptual schema, the external schema hides all parts of a database a user is not interested in or is not allowed to see for security reasons.

Several external schemata derived from the same conceptual schema can coexist to cater to the needs of different users or groups of users, regardless if the user is actually a person or an application using the services of the database.

These schemata are merely descriptions of data, the only data that actually exists is stored at the physical level.

If an user or an application makes a request, the request has to be transformed first from a request made at the external level to a request made at the conceptual level and then the request made at the conceptual level has to be transformed to a request at the internal level. The reverse has to be done for the result of the request. These transformations are called *mappings*.

Mappings allow changes at a given level of the framework to be absorbed (to a certain degree) without affecting other levels. This capacity to change the schema at one level of a database without having to change the schema at the next higher level is called data independence. Two types of data independence can be distinguished: logical data independence and physical data independence.

4.1.2 Logical Data Independence

Logical data independence implies that changes in the conceptual schema do not cause changes in external schemata or in application programs. There are two basic types of changes:

expansion: new data types are added to the conceptual schema. Expansions never cause changes to external schemata.

reduction: entity types are removed from the conceptual schema. Reductions cause changes only in those external schemata which contained the removed entities.

4.1.3 Physical Data Independence

Changes in the internal schema do not cause changes in the conceptual schema and thus also no changes in the external schemata.

Typical changes to the internal schema include file reorganizations and instantiation of additional access structures, which allow to fine tune the database for performance improvement. Changes to the internal-to-conceptual mapping allow application programs or users to automatically benefit from the changes to the internal schema.

4.2 Data Independence in Hypermedia DBMS

In this section relevant hypermedia data models and their implementations are examined. We show how they address the problem of physical and logical data independence and which constructs presented in the data models can be used to achieve data independence.

The Dexter hypertext reference model [Halasz and Schwartz, 1994] features three layers: The run-time layer deals with issues of presentation and user interaction, the storage layer maintains a database which contains a network of links and nodes, and the within-component layer deals with the structure and the content inside the nodes. The run-time layer can be interpreted as the external level, the storage layer as the conceptual level, and the within-component layer as the internal level of a database management system. To avoid confusion we will use the terms "physical level", "conceptual level", and "external level" in the following discussion instead of the terms "within-component layer", "storage layer", and "run-time layer" used in [Halasz and Schwartz, 1994]. Note: while we use the terms as used in the database community, the Dexter hypertext reference model does not describe a database management system. The functionality of the layers can be implemented in an unlimited number of independent applications.

The Dexter model is an excellent tool to model a hypertext application, yet there are some shortcomings if the model is applied to a hypermedia database management system because neither physical nor logical data independence can be achieved:

- Most issues (disregarding the interfaces of anchors described by the model) of the physical level (within-component layer) are outside the scope of the model, thus insulating it from specifics of the storage of these components. This abstraction is appropriate for a model describing an application, but it cuts out a vital and very interesting part of a hypermedia database management system, the storage system.

- Physical data independence is not addressed within the model. The mapping from the conceptual level (central storage layer) to the physical level is very flexible: The resolution of an anchor, which yields one specific piece of multimedia data, is delegated to the (possibly external) application handling the multimedia data. Thus the content of the multimedia data can change to a certain extent without affecting the conceptual level, at least if no anchors are deleted. However, the fixed association of one component in the conceptual level with one component (or document) in the physical level causes changes in the conceptual level if the internal structure (in contrast to the content) of the multimedia data changes. For example, if a video is split into three segments the anchors in the conceptual level have to be adjusted. Even more, the structure of the anchors has to be changed if the split of the video occurs inside the extent referenced by an anchor. Therefore physical data independence is not achieved.

- Logical data independence is not addressed within the model. The mapping from the conceptual to the external level (run-time layer) consists of presentation specifications embedded in the links. Changes in the conceptual level inevitably cause changes not only in the definition of the external level, but also in the external level itself. Applications which access this external level must be modified as well. Therefore logical data independence is not achieved.

  While the one to one mapping described in the Dexter model is sufficient for the model of an application which describes one hypertext network, additional constructs for the "trimming" and restructuring of the schema according to the needs of several different applications are desirable for a hypermedia database management system data model.

The observations made for the Dexter model also apply to varying degrees to the models discussed below.

In [Campbell and Goodman, 1988] HAM, a transaction-based server for hypertext storage, is described. HAM, the Hypertext Abstract Machine, plays the role of the conceptual level. Physical data independence cannot be achieved because the mapping into the physical level (the host file system) cannot be changed and thus does not allow alternative ways of storing data. While there is a mechanism to extract certain parts of the network of nodes and links based on the value of attributes attached to nodes, links, or contexts, the responsibility of dealing with this mechanism is offloaded to the applications using the server and thus logical data independence cannot be guaranteed.

Schnase et al. [Schnase *et al.*, 1993] implemented the hyperbase management system HB1. HB1 uses a sophisticated data model to describe the entities the hyperbase is constructed of (media objects, anchors, and links) and achieves a very high degree of interapplication linking. There is little indication how the actual hypermedia data is modeled beyond the usage of links. The physical level is straightforward (links, whose type is stored as an attribute, and media objects) and there seems to be no choice how to map the conceptual level to the physical level, thus physical data independence cannot be achieved. An external level does not exist, therefore the problem of logical data independence cannot be addressed within HB1.

Woelk et al. [Woelk *et al.*, 1986] describe an object-oriented data model for multimedia data which maps the semantically rich conceptual schema into a non first normal form (see [Sheck and Scholl, 1984]) relational schema. There is no provision for a different mapping into the physical level and there is no external level, thus neither physical nor logical data independence is achieved.

In [Kacmar and Leggett, 1991] PROXHY, a process-oriented extensible hypertext architecture consisting of a straightforward link and node hypertext data model is described. Layered components are used to model applications and their communication rather than the data, thus data independence is not achieved.

In [Gibbs *et al.*, 1994] three levels are described: (1) a BLOB (binary large object) is interpreted, (2) the resulting media object can be manipulated yielding derivations, which in turn (3) can be composed into multimedia objects. The process of interpretation can be compared to the physical-to-conceptual mapping, but there are no provisions for secondary access structures. There can be many media objects for one BLOB, but there are no provisions for a media object consisting of many BLOBs at the level of the interpretation. The concept of "derivation" is defined as "the mapping from a set of media objects, and a set of parameters that govern the derivation, to a new media object". This concept is regarded as a simple viewing mechanism, which is contradicted by the authors claim that "derivation provides physical data independence by separating derived objects from the underlying stored data". While a certain degree of physical data independence can be achieved for some technical aspects of multimedia programming, the model presented in [Gibbs *et al.*, 1994] does not deal with secondary access structures. Further, the model is not sufficient for hypermedia database management systems because it does not deal with links.

HyperBase [Wiil, 1993] implements a hypertext server which also features no distinction between the physical and conceptual level. An interface to the applications using the server is defined, but this interface does not protect applications from changes to the structure of the data.

In the hypergraph based data model described in [Tompa, 1989] views can be created based on the base hypergraph or other views by adding new hyperedges or also new nodes. While the model provides a mechanism for the creation of views, it does not achieve a clear separation of application specific knowledge and general knowledge, especially because new nodes (and thus new media data) can be added to views and thus not the whole structure and meaning of the media data of a hypermedia database management system is modeled in the conceptual schema. The problem of physical data independence is addressed through a function mapping the nodes of the hypergraph to the media content. Real physical data independence is not achieved because no alternative representations of a media object can exist.

The MD data model [Gu and Neuhold, 1993] contains a schema describing the conceptual, logical, and layout structure of a multimedia document. The conceptual structure describes the semantic relationship between the objects comprising the multimedia document. The logical structure describes how the content is divided, for example in chapters or sections. The layout structure models the layout of the multimedia documents, the specification of position and extent of their occurrence in space and time. While there are provisions to use data in different contexts, neither physical nor logical data independence can be achieved because all three kinds of structures are described in *one* schema in a tightly coupled manner.

The VIMSYS data model [Gupta *et al.*, 1991] describes a layered data model for the storage of images. The model provides sophisticated support for the storage of information derived directly from the image and of information entered manually. Queries based on these informations are supported as well. For one image several variations can be stored, but all variations must be derivable from *any* other variation of the image object. HyperD requires only that each variation must be derivable from one primary media object, because often variations have to be stored which contain less information than the original object, e.g. an image with less colors.

The HyperPro system [Østerbye and Nørmark, 1994] implements views to separate an internal hypertext representation from the external screen representation. This system is intended for rigidly structured domains such as source code and documentation. All links and nodes are typed (see Section 2.1.4.1 on the meaning of the term "type" in this context), views are generated by rules and seem to allow the selection of links of a certain type. There is no mechanism to restructure the hypertext network, with the exception of the possibility to include several (sub-)nodes into one node.

The system presented in [Gardarin and Yoon, 1995] features a general link and node data model, where only nodes can be subclassed. There is no specific support for modeling temporal or other kinds of multimedia-specific relationships that are supported by the HyperD architecture. A view in Gardarin and Yoon's system is a class of nodes, generated by a single HyOQL [Gardarin and Yoon, 1994] query.

In CHIMERA [Anderson *et al.*, 1994] views consist of a combination of an object and a viewer. Each view has an associated set of anchors and links, which are sets of anchors. Anchors cannot be shared between views. In this architecture there is no separation between general knowledge and application specific knowledge and thus no controlled sharing of the information expressed by application independent links.

One way to create views for nodes is described in [Speegle, 1995], where the emphasis is placed on reducing storage space by storing the program (in terms of steps performed by an editor) rather than the resulting node. While it is possible to combine several nodes into one resulting node, there is no provision for dealing with aspects of hiding or creating of links or of storing or updating of materialized views. The functionality described by [Speegle, 1995] is hidden in the storage level of our architecture, because the decision if an image is stored as an image or as a query resulting in an image is not application specific.

OOHDM [Schwabe *et al.*, 1996] (see also section 3.2.3.17) also offers a view mechanism: the navigational class schema defines which entities of the conceptual schema are visible to the user and how they are mapped into pre-defined navigational classes like nodes, links, or access structure. The main goal of this mechanism is not to ensure logical data independence, but to map the entities of the conceptual schema to simple data objects, which can be dealt with by a wide variety of hypermedia platforms. Thus the mapping from the conceptual schema to the navigational schema does not take the way the data is used into account (the "application of the data"), but only the application used to *display* the data. Since the mapping is fixed (for a specific application used to display the data), changes in the conceptual schema cause changes in the navigational schema, and thus logical data independence is not achieved.

Chapter 5

HyperD Architecture

This chapter contains a detailed description of the **Hyper**media **D**atabase Management System Architecture, of its schemata and of the classes of the schemata. A high level object model of the HyperD architecture is shown in Figure 5.1. The figure shows only selected classes to give a broad overview of how the schemata are related. Note: neither attributes nor methods are shown in Figure 5.1. The full object models are shown and discussed in Sections 5.2 to 5.5.

Some aspects of the HyperD architecture, especially those regarding the implementation of selected parts of the architecture, are discussed in Chapter 8. Certain aspects regarding the interface between the levels and a discussion of data independence in the HyperD architecture can be found in Chapter 6.

This chapter begins with an introductory section, which contains some notes about the way the work is presented in this chapter as well as underlying assumptions. Sections 5.2 to 5.5 each discuss one level of the HyperD architecture, starting in a bottom-up manner with the storage level and ending with the presentation level.

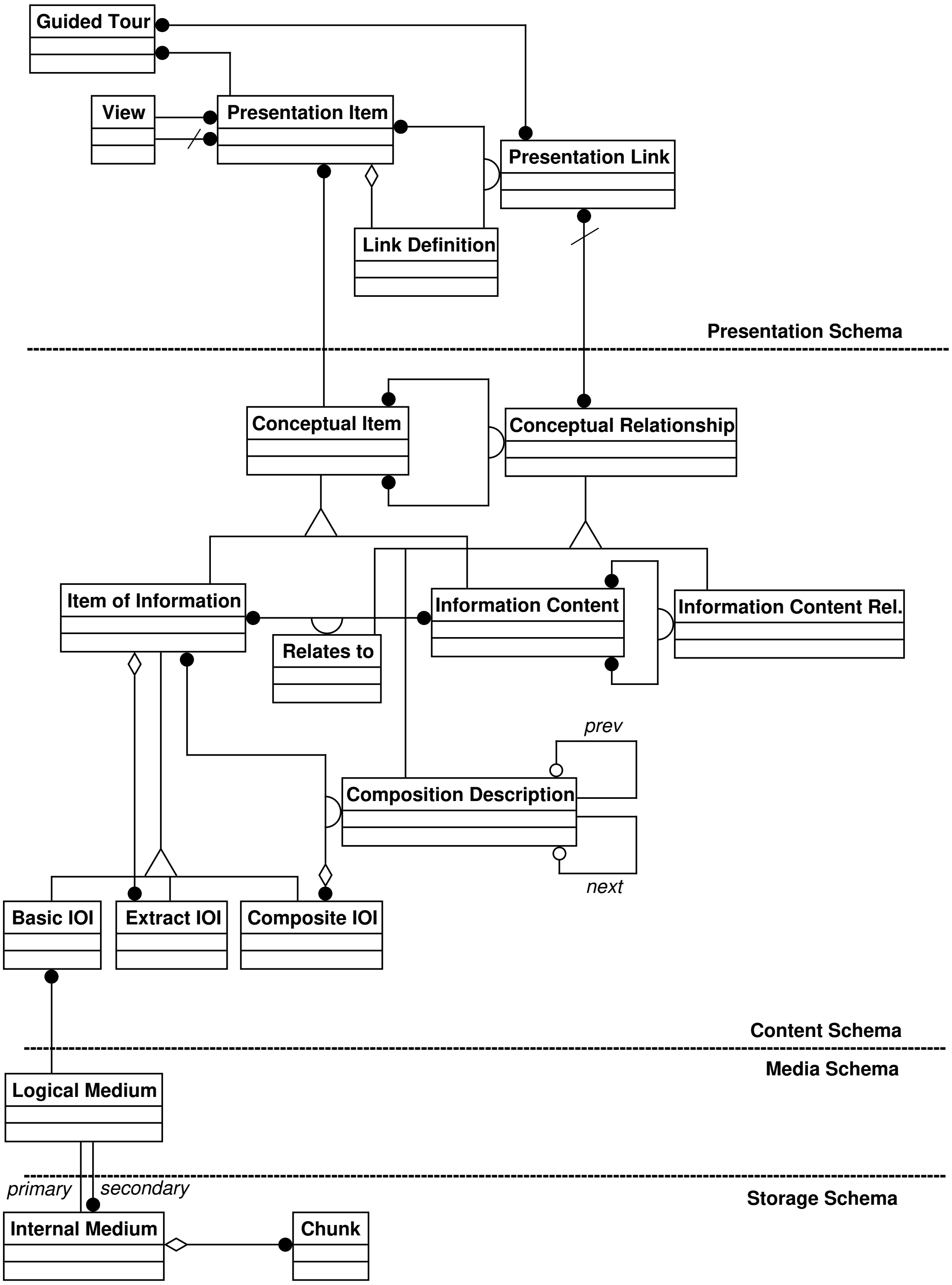

Figure 5.1: High Level Object Model

5.1 Clarifications and Remarks

This section contains clarifications and remarks about the way the HyperD architecture is presented in this chapter.

Objects are instances of exactly one class. Objects always belong to the most specific class (i.e. the leaves of inheritance hierarchies), all other classes are abstract classes. One consequence of this rule is that classes with existing instances cannot be subclassed anymore.

The instances of the classes shown in the figures of this section are regular objects, *not* classes. The only exception is Figure 5.15—a meta class model—which shows meta classes whose instances are classes.

The architecture consists of four levels, the objects of each level are described by a corresponding schema. At the level, the purpose of the objects contained in said level is described, while in the schema the structure and relationships of the objects is described.

In the work presented here, it is not attempted to present a "complete" architecture, but a framework for such an architecture. While all classes of the architecture are presented and discussed in this thesis, the classes are incomplete in that the attributes and methods of the architecture are not fully specified. The methods and attributes shown in the figures of this chapter are used to illustrate key concepts of the architecture and are to be considered as examples. They are *not* a complete list of methods and attributes. Implementation dependent attributes and methods have not been included in the figures. Operations for creating, modifying, and deleting instances as well as operations to establish or remove relationships are also not shown in the figures of this chapter. For a detailed description of the levels see the corresponding sections in this chapter.

The architecture was constructed adhering to the principles of object oriented programming, one being information hiding. In the object models, the methods for accessing and modifying the attributes are not shown with the intention not to clutter the model unnecessarily. For each attribute, a "get" method to retrieve the value and a "set" method to set the value of the attribute have to be implemented.

That the classes of the HyperD architecture are sufficient to build a hypermedia database management system is proven by construction in a related master thesis [Lisberger, 1998]. In the master thesis, supervised by the author of this thesis, the classes and concepts presented in this thesis are used to implement a hypermedia database management system using the object-relational database management system Illustra. The implemented system is limited to images and text, yet illustrates that the concepts presented here can be implemented successfully and with the classes presented here. The master student decided to implement some aspects in a different, simpler, and more restrictive manner to save implementation time. For example, the query optimization is implemented for only a few operators and it is implemented in a very simple, unflexible manner which cannot be extended to cope with all operators, as it is described in this thesis. Since the prototype is limited to images and text, neither temporal composition nor propagation of relationships is implemented. As of the writing of this thesis, the implementation has been completed, while the master thesis is being written.

5.2 Storage Level

The storage level provides information about the actual storage structure of the multimedia data. Information stored in this level includes for example the number of chunks a multimedia object is composed of, its storage location, its storage format, its sampling rate, and the multimedia data itself. In addition to the data inserted into the database, secondary storage structures can be created to accelerate query processing.

The object model of the storage schema is shown in Figure 5.2.

This section first describes the classes of the storage schema, Chunk and Internal Medium, and then describes how the internal media are connected to the objects of the media level.

5.2.1 Chunk

Instances of subclasses of class Chunk store one piece of multimedia data each as well as the technical information (in contrast to semantic information) necessary to manipulate and to play the multimedia data to the user. This information includes the storage format, the location of the chunk and media dependent attributes like for example the size for images and the length and size of videos.

At the bottom of Figure 5.2 subclasses of class Chunk for audio, video, images, and text data are shown.

5.2.2 InternalMedium

An InternalMedium is a nonempty collection of Chunks and represents one variation of a LogicalMedium (see Section 5.3 for details on logical media). An InternalMedium is either a primary or a secondary variation of the LogicalMedium: the primary InternalMedium consists of the chunks which were originally inserted into the database to be represented by the corresponding LogicalMedium, a secondary InternalMedium is derived from the primary InternalMedium to accelerate query processing.

In the context of this thesis, a variation is any version of the original media data which has been modified in a certain way. Such modifications can be selections of certain segments of the data, usage of different encoding techniques, and certain transformations (for images, e.g. gamma correction and effect filter). We have chosen the term "variation" to contrast it to "derivation" as used, e.g., in [Gibbs *et al.*, 1994] (see Section 3.1 for a detailed discussion).

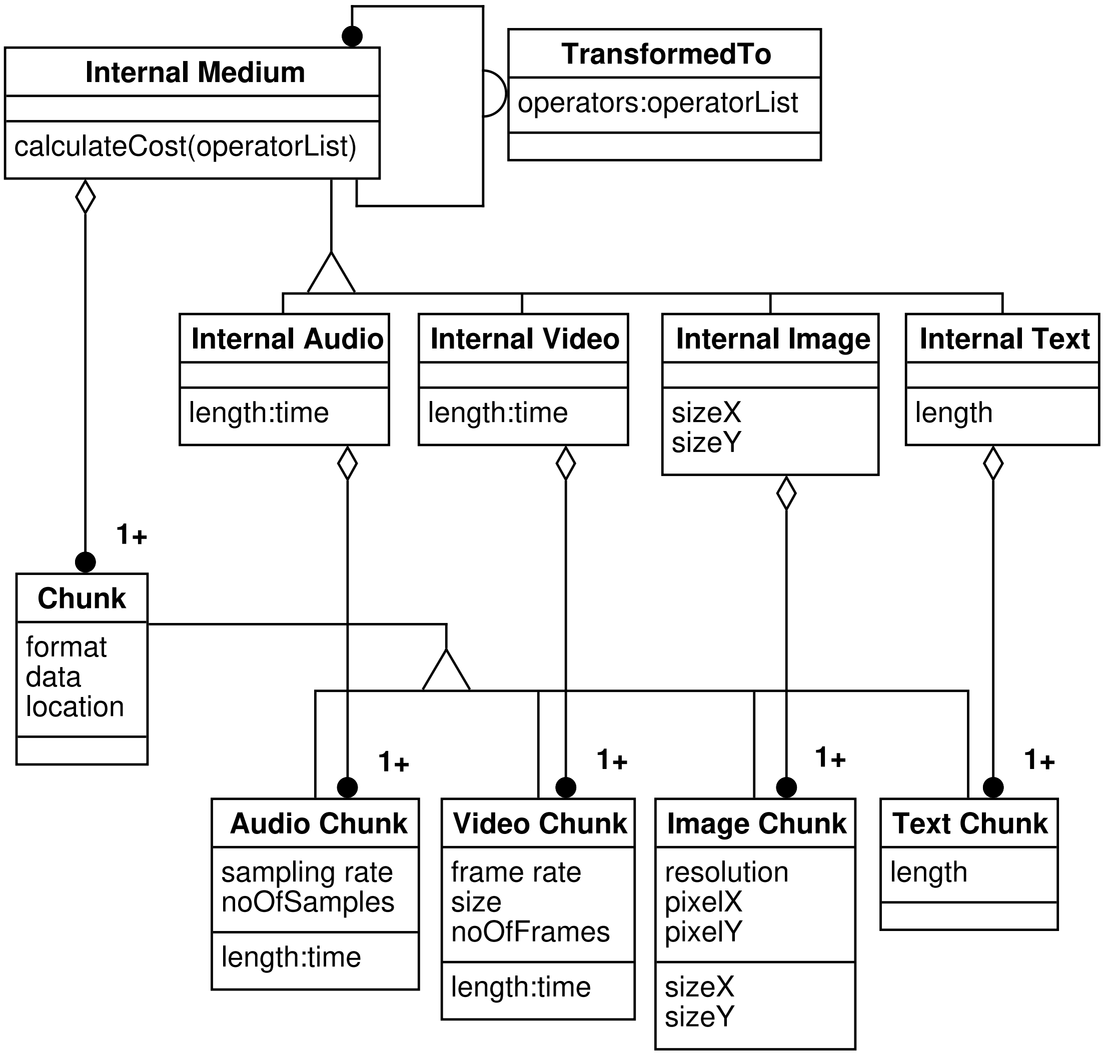

Figure 5.2: Object Model of the Storage Schema

Each internal medium can consist of any number of media chunks, thus the decision whether a certain video is stored as one piece or whether the video consists of three parts distributed on different disks, is hidden from the higher levels of the database. The relationship between the internal medium and its chunks is an ordered aggregation, the interpretation of the order of the chunk is the responsibility of the internal medium. For example, the audio chunks belonging to an InternalAudio are concatenated sequentially. The aggregation relationship between chunks and the internal medium states that the existence of the chunks depends on the existence of the internal medium.

5.2.3 Relationship between Internal Media and Logical Media

When media data is inserted into the database, a LogicalMedium is created. One exception to this rule is described below: media data can also be inserted as data for a secondary internal medium. Each logical medium has exactly one primary internal medium, which is also created on insertion (see also Figure 5.4). The chunks belonging to the primary data contain the actual media data originally inserted into the database to be represented by the corresponding LogicalMedium.

In addition to the primary internal medium each logical medium can have an unlimited number of secondary internal media, which are derived from the primary internal medium or from another secondary internal medium. The internal media belonging to a logical medium form a directed, acyclic graph starting at the primary internal medium. The edges of the graph are modeled by instances of the relationship class TransformedTo, which also store the operators which led to the creation of the secondary internal medium.

The chunks belonging to a secondary InternalMedium are derived by applying operations like merge/extract or other media-dependent operations to the chunks of existing internal media. For examples and a discussion see Chapter 8.

Secondary internal media do not have to be derived by the database, the user can provide an already existing variation of the primary internal medium. This can lead to a higher quality of the secondary internal medium, because some operations, especially encoding into or decoding from a lossy image format (e.g. jpeg), can lead to a lower quality of the resulting secondary internal medium. This is done using method addSpecificSecondary of class LogicalMedium (see the following section).

Once media data is inserted into the database, its primary internal medium must not be changed, because it is (a) the starting point for the graph of secondary internal media as well as (b) the reference point for queries. For this reason, the primary internal medium cannot be replaced by a secondary internal medium, even if the primary internal medium is no longer used itself. When the primary internal medium is replaced, the network of secondaries would have to be recomputed and in some cases, especially if the new primary internal medium is only a part of the original primary internal medium or contains for any other reason less information than the primary internal medium, this reconfiguration is not possible. This policy of the fixed primary internal medium in some cases leads to an increase in the storage space needed, but has no consequences whatsoever on the performance.

The lower half of Figure 5.3 shows an example: A large number of display devices can display 255 colors while the image of this primary internal medium can contain up to 16.777.216 colors (3 byte per pixel). So the secondary internal medium S0 is created by applying the operator "quant(255)" to the image of the primary internal medium (This operation lasted 40 seconds on a typical Sparc 4). Because the image is often needed in a smaller size, 640x480 pixel instead of 1200x900, the secondary internal medium S1 is generated (This operation lasts 11 seconds). Finally S2 is created, which shows Jack with a margin around the image. These secondaries are used to speed up the execution of queries sent to the logical medium. For more examples and a description of how these internal media are used see Chapter 8.

As shown in the example, some internal media correspond directly to certain

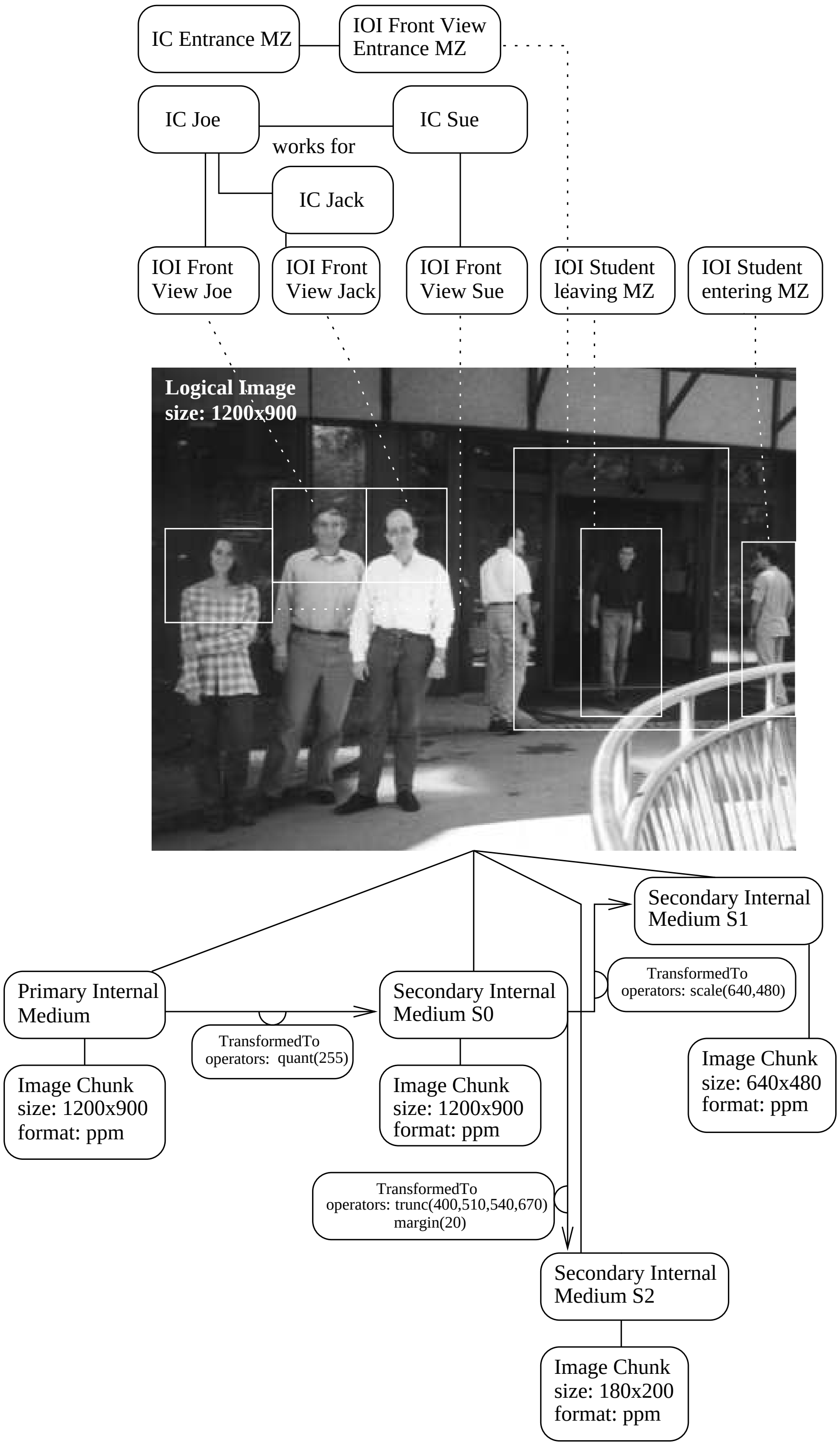

Figure 5.3: Example Internal Media

IOIs. For example, the secondary internal medium S2 corresponds to "IOI Front View Jack". Still, there is no direct connection from the IOI to the secondary internal medium, because the information if the secondary internal medium corresponds to an IOI and, if so, to which IOI is not relevant. For example, if the IOI is deleted, this does not necessarily mean that the secondary internal medium should be deleted too, because it can be used to accelerate other queries as well.

5.3 Media Level

The media level provides information about multimedia data disregarding the storage structure. Multimedia data is described in the media schema by instances of a subclass of Logical Medium. The object model of the media schema is shown in Figure 5.4.

This section describes the objects of the media schema, i.e. instances of classes derived from Logical Medium, some considerations about the way knowledge is distributed over the schemata, and it also discusses the benefits of this approach to the physical storage of multimedia data.

5.3.1 Logical Medium

The logical media objects in the media level serve as the abstract representation of the multimedia data stored in the database. Whenever a piece of multimedia data is inserted into the database, except to supply media data for a secondary internal medium as discussed above, exactly one logical medium is created together with a primary internal medium.

At this level only some basic properties of multimedia data are known, e.g. the length of a video recording, most other properties are hidden in the storage level. Logical media are responsible for query processing, query optimization,

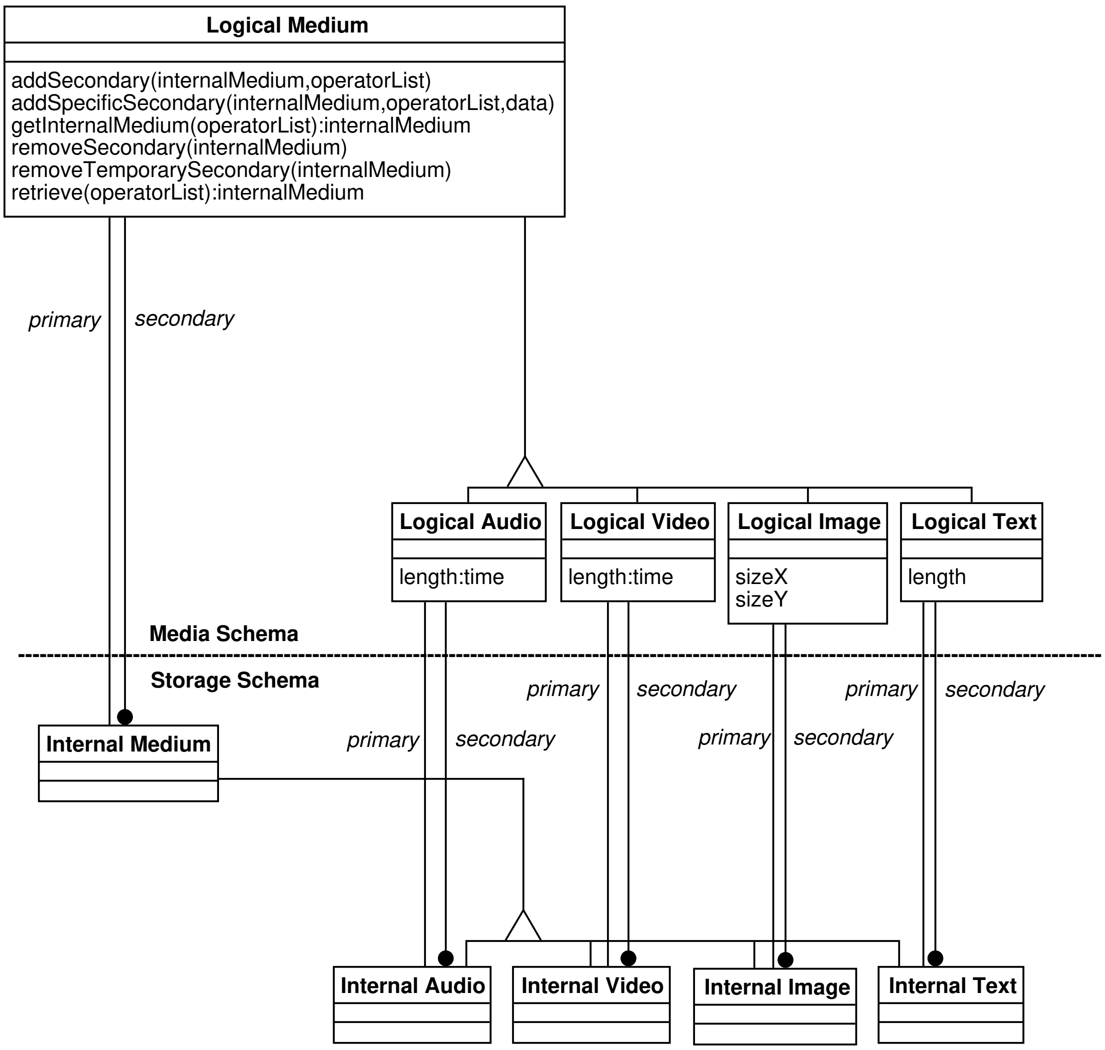

Figure 5.4: Object Model of the Media Schema

and for the management of the storage level, e.g. for the instantiation and for the deletion of secondary internal media.

When a request is handed down from the presentation level and the content level, the objects of each level add operators to the query which yields a set of operators which is passed in turn to the logical medium. The content level, for example, adds the operators which extract the desired part of the multimedia object. The logical medium calculates the most efficient way to fulfill the request and returns a concrete representation of the multimedia object, e.g. an image of a certain specified size (see also Chapter 8).

5.3.2 Design Considerations

One question is whether the original, external storage media of the media data, e.g. CDs, audio or video cassettes, should be modeled, if desired, in the content schema or as special classes in the media schema, e.g. Audio-CD. We have decided that entities like CDs, audio and video cassettes are to be modeled in the content schema, but not in the media schema. This decision is based on the discussions presented in this section.

Regarding external storage media, two kinds of structures can be distinguished:

- Technical structure refers to the placement of the tracks on the CD. This information is, if necessary at all, captured at the lower level of media chunks, because it represents information about the physical storage structure.

  On the level of logical media it is not relevant in which order or which combination the tracks have been stored on a CD or a set of CDs. It is also not relevant if the track is stored on a CD at all. These considerations can also be applied to other storage media, e.g. laserdisks or video cassettes. Therefore, we do not model CD's or similar entities in the media schema.

  The logical medium abstracts from details of the physical storage, thus modeling Logical CDs for the sake of capturing the technical structure would add information not relevant in the media schema.

- "Semantic" structure: this type of structure describes how the tracks on the CD are related to each other. The (sequential) placement of the tracks on the CD can describe the sequence of the movements of a concert or a historical ordering of the recorded tracks. On other CDs the grouping is done on the stylistic era or even the mood of the piece. The tracks on CDs can share the same interpret, conductor, place of recording or the fact that the tracks hit the charts in the same year.

  Capturing this information on the level of logical media would result in a large number of subtypes of CDs, which would enforce a rigid type of classification which has to be adjusted for different purposes of usage.

  The information about the semantic structure can be modeled conveniently in the content schema. The CD can be modeled as an IC-Object and the information is embedded seamlessly in the knowledge structure without the need for duplicating this information on the level of logical media.

The discussion presented above is a part of the question which meta information should be known about a logical medium. For the purpose of this discussion, meta information can be grouped into two categories:

Technical information: this kind of information describes properties concerning the multimedia data itself, but *not* the content presented by the data. Examples are the resolution of images or encoding of audio recordings.

This information is stored at the storage level.

Non-Technical information: this is basically everything which is not a technical information. Examples are photographer, date, and location for an image or producer, interpret, and orchestra for an audio recording.

For this type of information, the same considerations as presented above for semantic structure can be applied: It is impossible for the programmer of the database to anticipate all possible kinds of information which users may want to have stored, for e.g. an image, even if a large number of subtypes of Logical Image is created. The same information can, if needed at all, be modeled conveniently in the content schema, where for each type of image (e.g. images of persons, buildings, landscapes, ...) a class can be defined which only stores the relevant attributes in a flexible way. Thus the responsibility of storing this meta information is shifted from the physical level (and the database programmer) to the conceptual level (and the database administrator), where it can be incorporated easily and seamlessly.

5.3.3 Combined Logical Medium

One possible extension, not shown in Figure 5.4, are Combined Logical Media. Combined Logical Media can be added to the HyperD architecture to provide a consistent interface for media types with an already fixed synchronization, e.g. a stream with interlaced video images and audio samples. Each channel can be extracted from the combined logical medium and then be attached to a basic IOI. The channels can be used together or selectively, e.g. combining channel 1 (video) with channel 3 (German audio).

5.3.4 Benefits

The physical storage of multimedia data as proposed here offers several advantages, for example:

Reuse of Multimedia Objects: Multimedia objects can be used in an unlimited number of applications without the need of redundant storage because the external view of the multimedia object can be calculated on the fly.

Performance: One conceptual picture is used by three applications, neither of which can use the picture directly but has to transform it. If one application has to be speeded up, the picture as needed by the application is stored as a secondary internal medium. Whenever the other two applications request the picture, the cost to transform the picture from the primary internal image and from the secondary internal images is computed and the cheapest variant is used. Thus each additional secondary internal medium is used automatically, if it accelerates the processing, without the need to update any application. If a secondary internal medium is deleted, the only consequence are possibly lower response times for certain queries.

5.4 Content Level

The content level provides information about the objects contained in and described by the multimedia data. Figure 5.5 shows the object model of the content schema. Note: classes Basic IOI and Composite IOI are shown in full detail in Figures 5.7 and 5.8.

The first section describes the two building blocks of the content schema: Information Content Objects and Items of Information. The section thereafter describes relationships between objects of the content schema. The final section discusses the role of whole-part relationships in the content level of the HyperD architecture.

5.4.1 Information Content Objects and Items of Information

The content schema uses Information Content Objects (IC-Objects, introduced in [Prückler and Schrefl, 1995]) to model explicitly some information that is contained implicitly in multimedia data. IC-Objects are also used to model real world objects not directly connected to multimedia data.

IC-Objects represent abstract entities of information. Concrete information

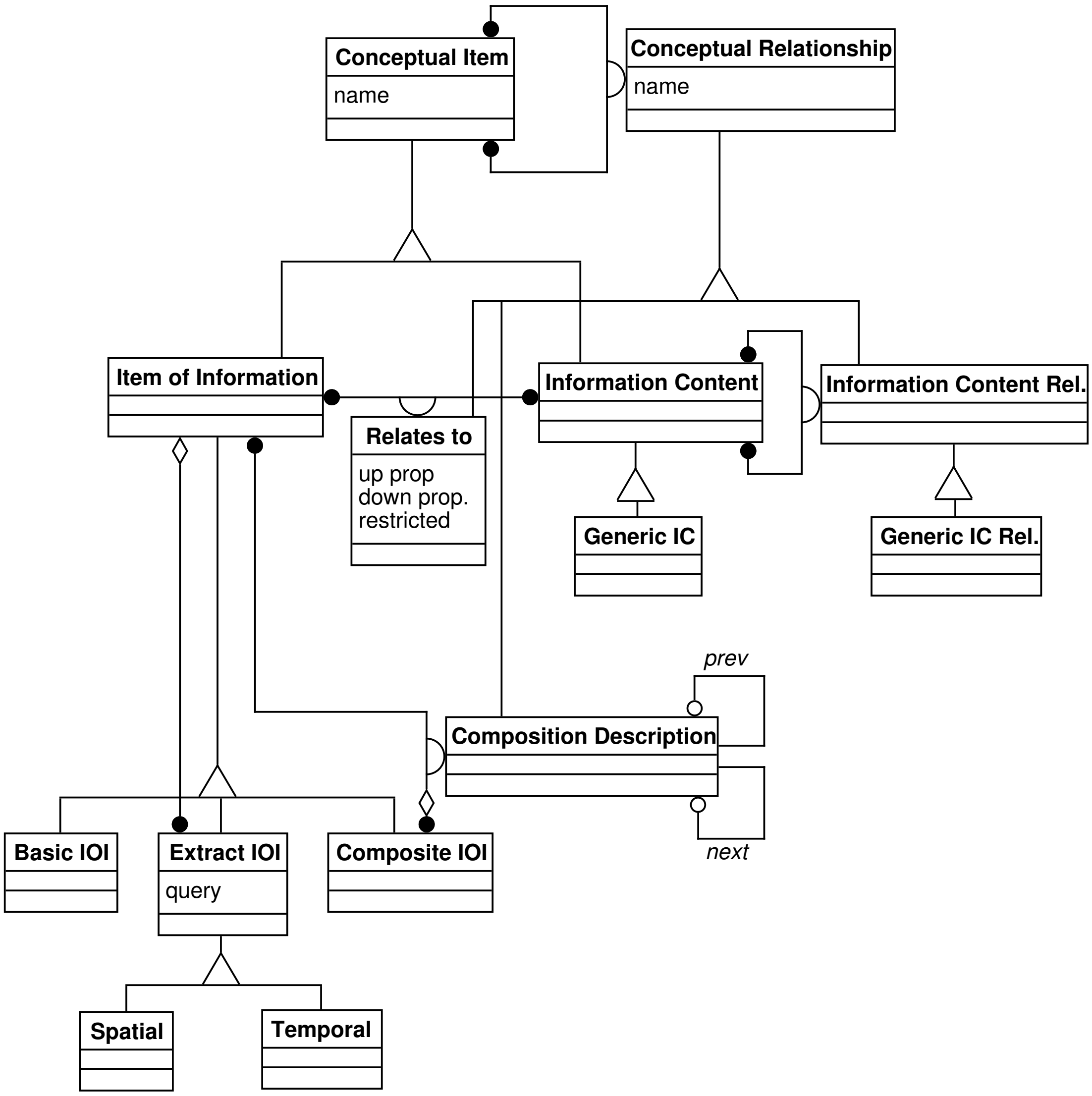

Figure 5.5: Object Model of the Content Schema

about an IC-Object are described by Items of Informations (IOI). IOIs describe the information which is conveyed to the user of the database by interpreting (e.g. viewing or reading) multimedia data.

Items of Information can overlap: different IOIs can refer to the same or overlapping parts of a logical medium and different IOIs can refer to similar information. As media artefacts can have different meanings to different persons or in different contexts, the point of view at which the artefact is looked at can change, as well as the IOI. In this way one media object can be used by a variety of people for a variety of purposes. For example, an image can, at the same time, convey more than one item of information: the bottom left image in Figure 5.6 shows "a building" *and* "a couple" standing in front of it.

Basic IOIs are the only objects of the content level which have a direct connection to the logical media representing the multimedia data (see Figure 5.7). In Figure 5.6, "IOI 1" to "IOI 5", "IOI G", "IOI E", and "IOI Rath." are basic IOIs. Other subclasses of IOI, Extract and Composite, are discussed in detail in the remainder of this section. For a discussion on the notation used in the examples, see Section 2.2. Objects "TC1" to "TC 9" are instances of class Temporal Composition (see Section 5.4.3.1).

IC-Objects as well as IOIs can be connected by relationships, thus forming a network of information. These relationships and the classes as shown in Figure 5.5 are described in detail in Sections 5.4.2 and 5.4.3.

As shown in Figure 5.6, ICs can be subclassed, thus allowing for easy extension of the core HyperD architecture to model domain-specific information. As stated in Section 5.1, only leaf classes of inheritance hierarchies may have instances. To be able to store objects for which no particular classes have been created, classes Generic Information Content and Generic Information Content Relationship have been added (see Figure 5.5).

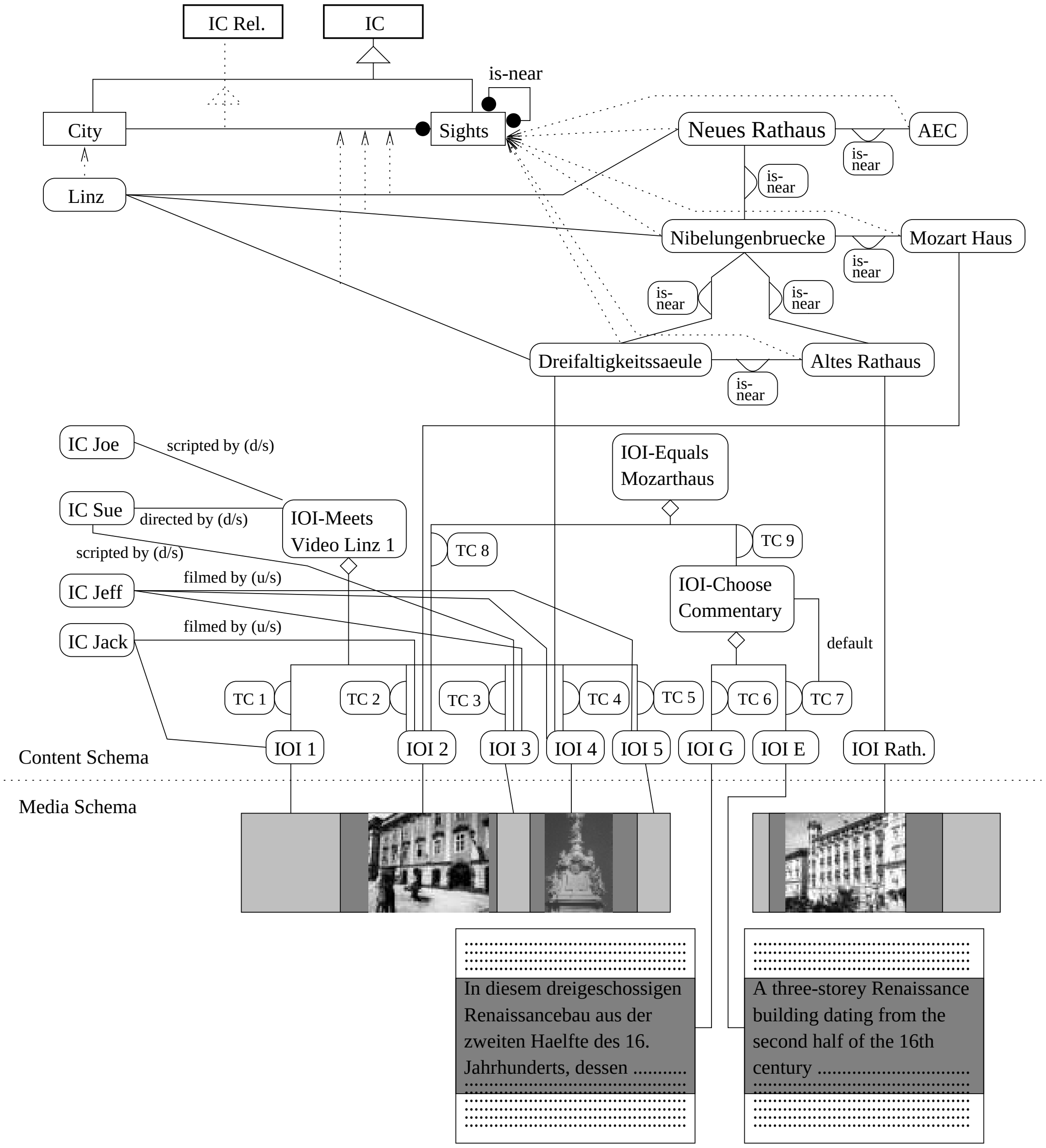

Figure 5.6: Example Content Schema

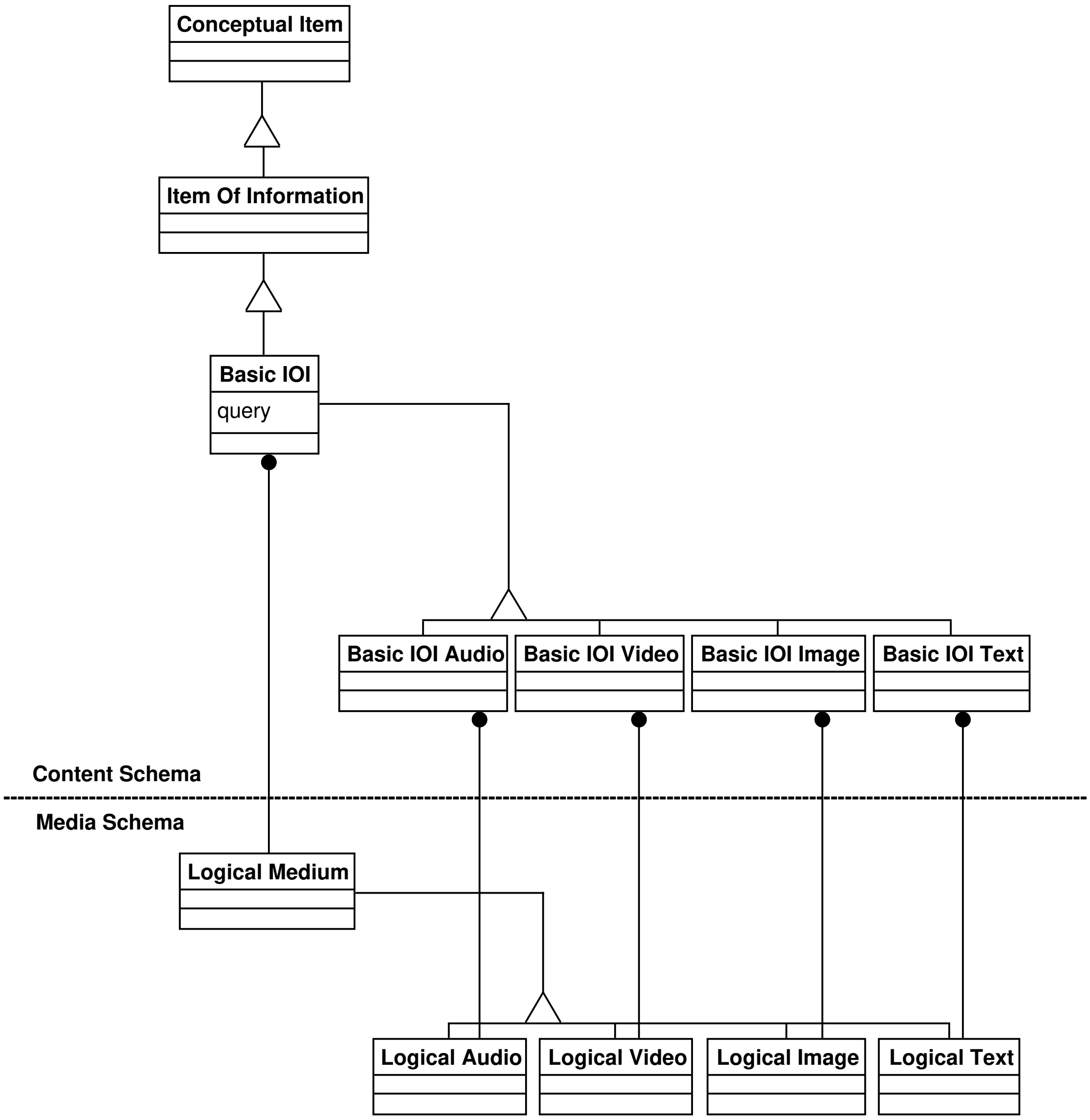

Figure 5.7: Detail from the Object Model of the Content Schema: Interface to the Media Schema

5.4.1.1 Differences between IOIs and ICs

IOIs and ICs are both derived from the same superclass, Conceptual Item. For the following reasons, ICs and IOIs are different concepts:

- ICs and IOIs have a different meaning: ICs store abstract knowledge and relate this knowledge to other ICs with relationships. ICs model the content, which is relevant or interesting for the user, while IOIs model the structure (in the sense of composition) of the media data.
- The separation of IOIs and ICs decouples the abstract information represented by the IC from the media data represented by the IOI which is used to convey this information to the user. This is especially valuable when media data changes, because these changes do not cause changes to the network of ICs, but only cause the reconnection of the IC to a different IOI. For further discussion see Section 6.2.
- IOIs and ICs have a different internal structure and also have different functionality. For example, while the media data represented by IOIs is displayed on the screen, the network of ICs is only visualized in a generic way while browsing. So IOIs and ICs are displayed in a different manner and thus need different methods.
- IOIs and ICs can be created independently, thus allowing both: top-down as well as bottom-up approaches to information modeling. Some types of IOIs can be created automatically, e.g. by detection of shot-boundaries in videos.

5.4.1.2 Extent of IOIs

The perceived content of multimedia data is highly subjective. There is no hard rule what extent of a multimedia object, as represented by a LogicalMedium, is an IOI. For example, there are no rules on which IOIs to create for the image shown in Figure 5.3, or on how big the regions represented by the IOIs should be. The decisions on which IOIs to create and how big to create them are conscious and subjective decisions of the creator of an IOI. As pointed out in the introduction, this is a inherent problem for hypermedia data. The decision is in fact the same decision as the decision about the content of a node, where also no rigid rules have emerged, beyond the timidly stated "A node represents one idea" [Conklin, 1987], which is hard to operationalize because different people see the "extent" of the same idea differently.

As no rules can be given, the architecture proposed in this thesis has been created to be highly flexible and to allow to model multimedia data and its content from many different points of view uniformly.

One could argue that this diminishes the value of the concept of an IOI. The act of creating an IOI can be compared to an act of communication: the creator wants to convey some meaning which the user of the media object and thus of the IOI is supposed to understand. So the situation is quite similar to a face-to-face communication, where the success of the communication is based on the factors of (a) a common background of experience, without which communication is not possible and (b) the context of the facts transmitted by the communication.

5.4.2 Relationships in the Content Level

This section describes four basic types of relationships between the classes of the content level.

IOI-Composite: The IOI-Composite relationship is based on composition (see Figure 5.8). Existing IOIs, which can in turn be composites, are combined into a new composite. A composite is the sum of its parts, i.e. there is no additional media data. Composites are used to model spatial as well as temporal composition. Spatial and temporal composition are applied in a similar manner: the whole is composed of an ordered list of parts, which in turn can be other composites, basic IOIs, or extracts (see below). How the order of the parts has to be interpreted depends on the class of the composite. In Figure 5.6, five segments ("IOI 1"–"IOI 5") are composed into the IOI "Video Linz 1". The class of this IOI is Meets, which means that the IOIs "IOI 1" to "IOI 5" are played sequentially.

See Section 5.4.3.1 for a detailed description.

IOI-Extract: The IOI-Extract relationship is based on decomposition. In contrast to the IOI-Composite relationship, where all components of an IOI constitute the IOI, all extracts of an IOI taken together need not comprise the whole IOI. Extracts allow to attach a certain meaning to some region of the IOI they are derived from, but extracts can also be used like any other IOI. Extracts support the natural modeling of clickable images and similar media objects.

See Section 5.4.3.2 for a detailed description.

IOI-IC: The IOI-IC relationship (shown as Relates To) relates the media data represented by an IOI to the abstract entity represented by an IC-object. Relates To relationships can be propagated upwards and/or downwards the composition hierarchy as propagated relationships. For example, using propagated relationships allows to state in an easy and consistent way that the director of a video also directed every part of the video. See Section 5.4.3.4 for a discussion of propagation of relationships.

IC-IC: IC-IC relationships (shown as Information Content Rel.) model semantic relationships between ICs. The class IC as well as the relationships Information Content Rel. and Relates to can be subclassed (subclassing of relationships is described e.g. in [Rumbaugh, 1993]).

The media level and the content level combined are similar to the SNITCH hypertext data model [Mayfield and Nicholas, 1993]: the links do not lead directly from one logical media object to another logical media object, but rather from (a part of) a logical media object to a semantic network depicting the relationships of the objects described by the multimedia data and finally back to another, not necessarily different logical media object. The main differences between our architecture and the SNITCH approach are our semantically richer network, that SNITCH supports only hypertext and no temporal relationships between the nodes, and the embedding in the larger context of the HyperD architecture.

5.4.3 Whole-Part Relationships

Component composition (aggregation) and decomposition are often considered as two sides of the same coin. This section shows that two types of whole-part relationships, one based on composition and the other based on decomposition can be used beneficially to model temporal and spatial relationships of multimedia data. The properties of these relationships are discussed and it is shown how they can be embedded into the content level of the HyperD architecture.

Whole-part relationships are of great importance for a wide variety of AI domains, e.g. robotics, vision, and natural language processing, and play a fundamental role in the modeling of information systems [Guarino *et al.*, 1996]: Whole-part relationships have been identified as an important modeling concept in semantic data modeling [Hull and King, 1987] and in object-oriented design notations [Rumbaugh *et al.*, 1991]. The importance of whole-part relationships is further emphasized by a special issue of [Data & Knowledge Engineering, 1996] "Modeling parts and wholes", which contains theoretical articles [Varzi, 1996; Smith, 1996] and a survey of whole-part relationships in object-centered systems [Artale *et al.*, 1996]. Whole-part relationships in object-oriented database management systems are described in [Halper *et al.*, 1992].

Whereas the articles cited above discuss whole-part relationships in general, we investigate the advantages of using whole-part relationships for modeling hypermedia data in particular. Until now, whole-part relationships are used in hypermedia and multimedia data models to model either temporal or structural/spatial composition, decomposition is not used at all (see Section 5.4.3.5 for an overview on the usage of whole-part relationships in existing hypermedia systems).

We will show that composition as well as decomposition are needed in hypermedia modeling to capture the full meaning of multimedia data, because using only one type of whole-part relationships would cause the loss of one important information: how something is created.

Important advantages of the approach presented in this thesis include an integrated approach to temporal and spatial modeling, a flexible and powerful way to incorporate multimedia data into a semantic hypermedia network, and an easier data management due to automatic handling of inserts and updates of dependent parts in whole-part relationships.

Whole-part relationships are used to model specific kinds of relationships between multimedia objects in the content level, which is shown in Figures 5.5 and 5.8. This section describes how the two types of whole-part relationships, composition and decomposition are used in the HyperD architecture and it discusses the properties of these relationships.

5.4.3.1 Composition: IOI-Composite

The IOI-Composite relationship is based on composition (see Figure 5.8). Existing IOIs, which can in turn be composites, are combined into a new composite. A composite is the sum of its parts, i.e. there is no additional media data. This is also called a *full* whole-part relationship in contrast to the partial whole-part relationship based on decomposition described in Section 5.4.3.2.

Spatial and temporal composition are applied in a similar manner: the whole is composed of an ordered list of parts, which in turn can be other composites, basic IOIs, or extracts (see Section 5.4.3.2). How the order of the parts has to be interpreted depends on the class of the composite. The remainder of this section discusses how temporal and spatial composition are incorporated into the HyperD architecture.

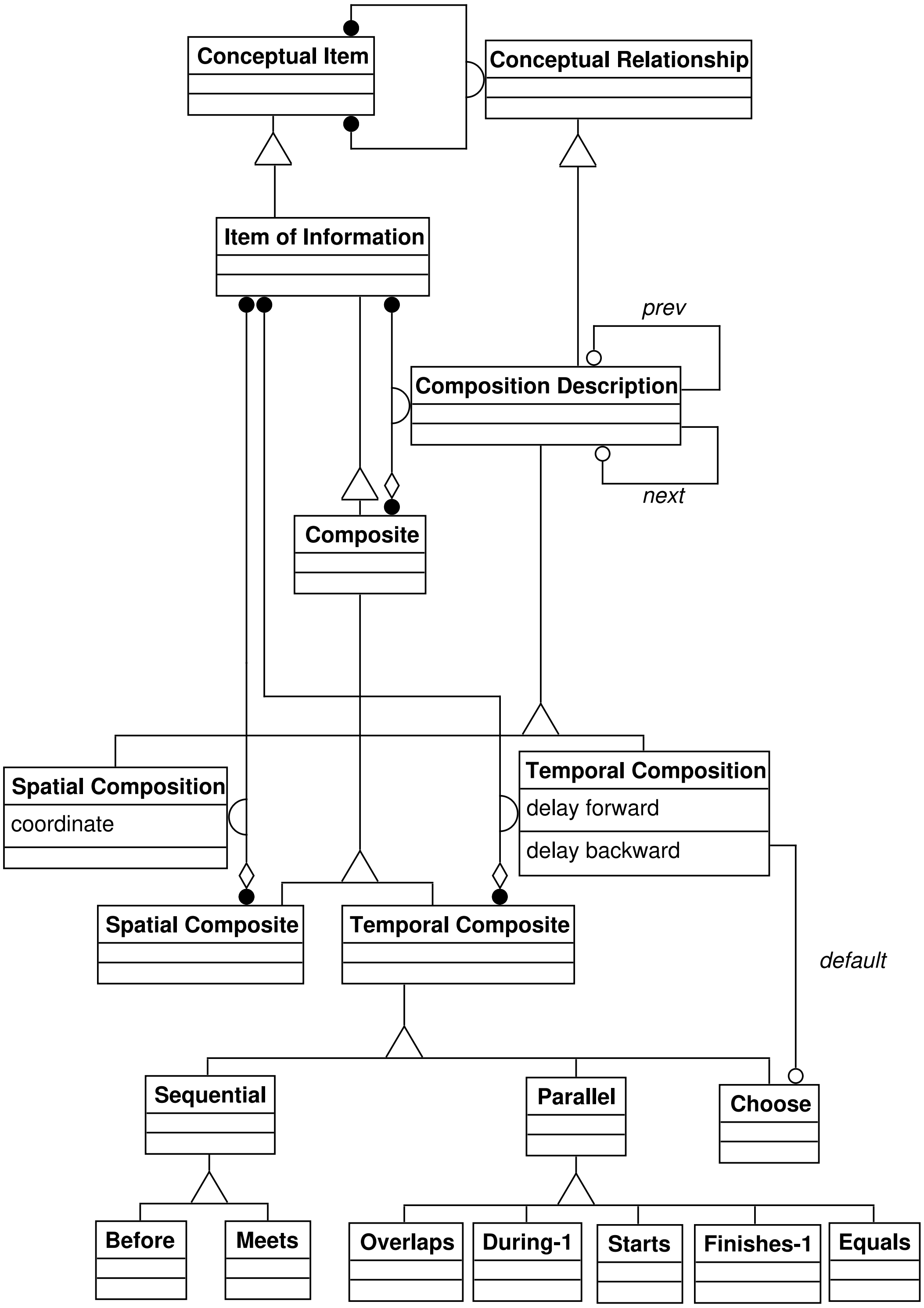

Figure 5.8: Detail from the Object Model of the Content Schema: Composition

Temporal Composition According to [Ackermann, 1994], three kinds of mechanisms for temporal specification can be distinguished:

Language specification: A program is used to specify a presentation. Examples are scripts [Dami *et al.*, 1988; Gibbs *et al.*, 1992; Vazirgiannis and Hatzopoulos, 1993; Vazirgiannis and Hatzopoulos, 1995], time stream Petri nets [Diaz and Sénac, 1994], object composition Petri nets [Prabhakaran and Raghavan, 1993], extended object composition Petri nets [Qazi *et al.*, 1993], or a multimedia event calculus [Schloss and Wynblatt, 1995].

Time line specification: The multimedia components are aligned along a time axis. This approach is used for example in MAEstro [Drapeau, 1993] and QuickTime [Davis and Burke, 1993].

Specification by hierarchical composition: A composition hierarchy is used to describe the temporal relationships between multimedia data objects. The type of the whole describes, using the relationships defined in [Allen, 1983], how the parts are related, e.g. if they start and/or stop at the same time or if they are played sequentially [Little and Ghafoor, 1993]. Comparable approaches are described in [Ackermann, 1994], where temporal relationships are modeled by sequential, timeline, looping, parallel, and delayed composition, in [van Rossum *et al.*, 1993], where synchronization is described by parallel or sequential relationships between multimedia data objects and synchronization arcs are used to model delay, and in [Buchanan and Zellweger, 1993; Buchanan and Zellweger, 1992], where a hybrid compiletime/runtime scheduling approach is described based on Allen intervals and higher-level constructs like grouping, iteration and conditional selection. Another model described in [Hamakawa and Rekimoto, 1994] is based on the concept of temporal glue. In [Layaïda and Keramane, 1995] an algorithm for maintaining temporal consistency for multimedia documents built with Allen-Intervals is presented.

The model described in [Adiba, 1995] falls in between language specification and hierarchical composition, the authors compare their work with script-based approaches, while the operators described in the paper are objects which are composed into a hierarchy. The additional meaning of a hierarchy is never used.

We decided to base our approach to temporal modeling on the work presented in [Little and Ghafoor, 1993], since it allows to express information about temporal relationships between parts explicitly in the schema rather than in variables of objects.

We distinguish between Sequential and Parallel temporal composition, the subclasses of these classes are named after the intervals described in [Little and Ghafoor, 1993] (see Figure 5.8 for the object model). Although the subclasses of Parallel and the subclasses of Sequential have the same attributes, they differ in the consistency constraints imposed on the parts.

The information how the IOIs of a composition are related is stored in relationship class Temporal Composition, which also contains the information how the starting time of the IOI is related to the staring time of the preceding/succeeding IOI of the same composition.

Synchronization of instances of subclasses of Parallel should be made as early, i.e. as low in the whole-part hierarchy, as possible. The reasons are that the information about the structure needs to be described only once and that the synchronization is more precise and easier to perform for smaller pieces. This rule is not meant to be restrictive, it is still possible to create alternative hierarchies should the need arise.

Temporal composition is extended by the Choose composition, which allows to select one IOI from a set of IOIs. For example, a video can be combined either with an English or with a German commentary (see Figure 5.6). If none is selected, the IOI is used, which is connected as the "part" in the whole-part relationship with the Temporal Composition pointed at by the default relationship. In the example, the English commentary is selected by default. This construct is similar to the "alternate groups" as described in [Davis and Burke, 1993], but it is not restricted to whole tracks and thus can be used at a finer granularity.

This construct is intended for choosing between several existing alternatives. The fact that a video either has a German, English, or French audio

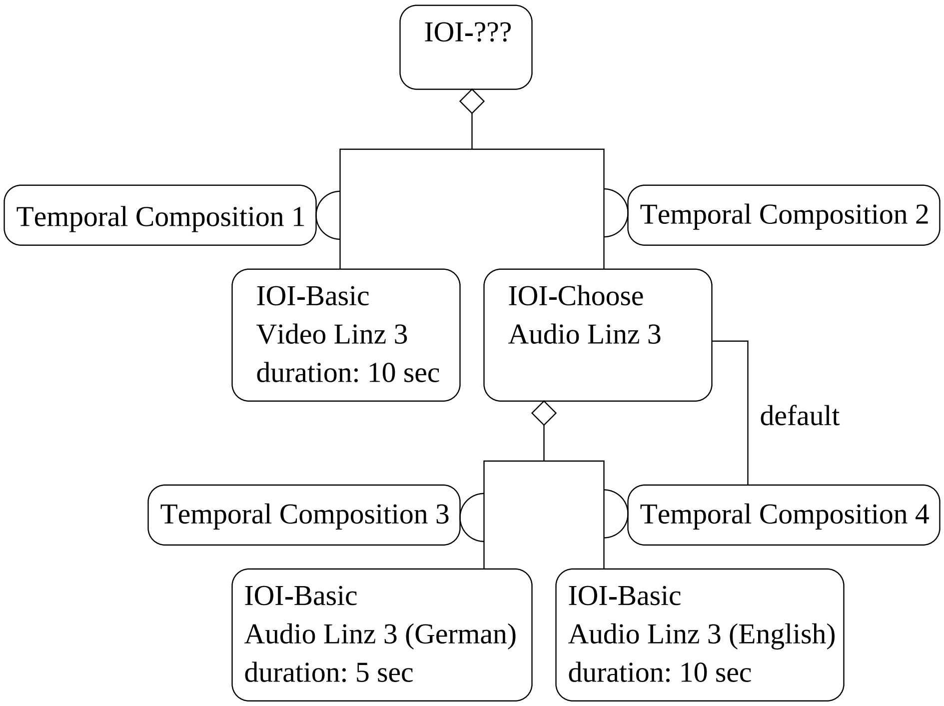

Figure 5.9: Example of a Temporal Inconsistency

(i.e. no Italian audio is allowed), has to be modeled with ICs using a is-a relationship.

All IOIs serving as component objects of a choose composite must have the same length. This restriction is necessary for class Choose, because IOIs of a different length could lead to temporal inconsistencies higher up in the IOI-Composite hierarchy, as shown in Figure 5.9: "Video Linz 3" has temporal relationship "Starts" to "Audio Linz 3 (German)", while "Audio Linz 3 (English)" has temporal relationship "Equals" to "Video Linz 3". Therefore, the class of the composite of "Video Linz 3" and "Choose" (shown as "IOI-???" in Figure 5.9) cannot be determined. Note: in the examples in this section the `prev` and `next` relationship occurrences are not shown with the intention not to clutter the examples unnecessarily.

Two examples of temporal composition are shown in Figure 5.6: the composite "Video Linz 1" is an instance of class Meets, so the IOIs "IOI 1" to "IOI 5" are played sequentially. The composite "Video Linz 1" models the video shown in the media schema in the content schema. Associated with the composite is the information about the people who worked on the video. Another example is shown to the right: the segment of the video showing the "Mozarthaus", "IOI 2", is combined with either an English or a German commentary. The object of class Choose states that either the English or the German commentary has to be chosen, which is then played simultaneously with the video segment, resulting in an object of class Equals.

Spatial Composition Two kinds of spatial composition can be distinguished:

- Spatial composition by arranging IOIs on the screen. This kind of spatial composition is application specific and can therefore be modeled, if desired, in the presentation schema by attributes of class Presentation Item (see Section 5.5).
- Spatial composition by combining IOIs into a new IOI. In contrast to temporal composition, spatial composition does not lend itself easily to a description based on a hierarchy of composites:
  - The instances, where a true hierarchy of several levels is created, are few. In most cases, one or more images are placed on a common background.
  - The number of operators used is rather large and varies with the type of media data which is composed.
  - There is not yet a clearly defined set of operators as it exists for temporal composition.

  For this reasons we have chosen to add only a simple spatial composition mechanism, which can be extended by inheritance should the need arise.

  When the composite is to be displayed, an empty image is created. Then one part after the other is pasted on top of the image. The size of the resulting image is determined by the "bounding box", i.e. the smallest box which fits around the resulting image. For example, the first page of a presentation of Linz is constructed by placing images taken from the videos on a map of Linz (not shown in Figure 5.6).

5.4.3.2 Decomposition: IOI-Extract

The IOI-Extract relationship (see Figure 5.5) is based on decomposition. In contrast to the IOI-Composite relationship, which is a full whole-part relationship, the IOI-Extract relationship is a *partial* relationship, i.e. the combined extracts do *not* have to add up to the complete IOI. There is no significant difference between spatial and temporal decomposition, thus the following discussion applies to spatial and temporal decomposition.

An Extract represents some part of an IOI and is derived from the IOI by decomposition. Extracts allow to attach a certain meaning to some region of the IOI they are derived from, but extracts can also be used like any other IOI. The query which led to the creation of an Extract is stored in the object and not in the relationship, because there can only be one query per extract. Extracts support the natural modeling of clickable images and similar media objects.

Some decompositions could also be modeled as a basic IOI, but this approach has a severe disadvantage: while basic IOIs can only reference media data from one logical medium, extracts can reference media data from more than one logical media, if the basic IOIs referring to the logical media have been spatially or temporally composed.

The name "Extract" was chosen instead of "Component" because this would have suggested that a component is always a part of a Composite. An Extract *can* be a part of Composite, but can also be a part of any other IOI.

As a design rule, we state that a new extract has to be created at the lowest possible IOI of the hierarchy considered. This rule is important because otherwise information about inferred whole-part relationships between IOIs can be lost as it is exemplified in Figure 5.10: Extract "Video Aerial View Danube" is derived from Composite "Video Linz 2", but is also a part of Extract "Video Aerial View Linz", which is derived from Composite "Video Introduction Linz". If Extract "Video Aerial View Danube" is derived from Composite "Video Linz 2" instead of Extract "Video Aerial View Linz", the information that Extract "Video Aerial

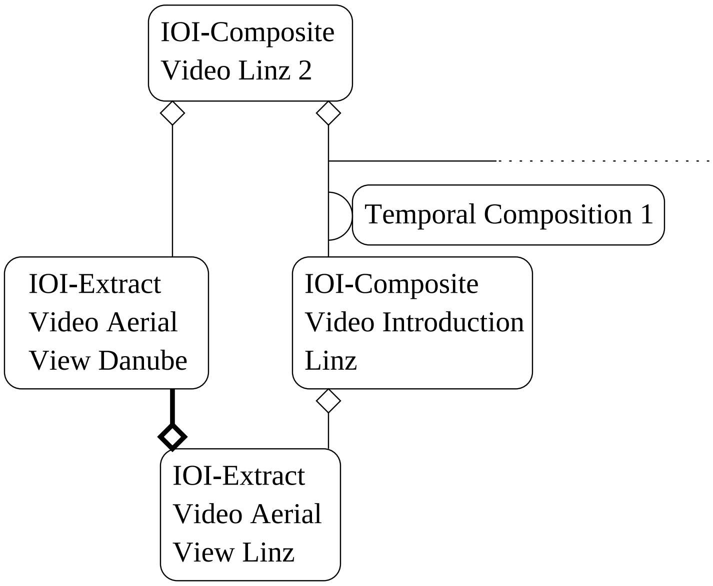

Figure 5.10: Placement of Extracts

View Danube" is a part of Extract "Video Aerial View Linz" (the relationship occurrence is drawn bold in Figure 5.10) is lost because it cannot be inferred from the relationship occurrence between Extract "Video Aerial View Linz" and Composite "Video Linz 2". If, however, Extract "Video Aerial View Danube" is inferred from Extract "Video Aerial View Linz", one can conclude that Extract "Video Aerial View Danube" is also a part of Composite "Video Linz 2".

5.4.3.3 Characteristics of Extract- and Composite-Relationships

In [Halper *et al.*, 1992] the characteristics of whole-part relationships are described. In this section we discuss these characteristics for the composition and the decomposition relationship described in the previous sections:

exclusive/shared: A part can belong to one whole *(exclusive)* or may belong to any number of wholes *(shared)*. Exclusive relationships can be subdivided into *interclass exclusive*, where the part belongs to exactly one whole, and *intraclass exclusive*, where the part belongs to exactly one whole of a given class, but at the same time may also belong to possibly many wholes of other classes.

The IOI-Extract relationship is exclusive with regard to the derivation relationship because one extract is derived from one specific IOI. Therefore, as discussed above, the query which led to the creation of the extract can be stored in the extract and does not have to be stored in the relationship to the whole-IOI. An extract can have whole-part relationships to other extracts (see also Section 5.4.3.2). These relationship occurrences are computed on demand and can be viewed by the user as part of the database schema.

The IOI-Composite relationship is shared, thus enabling reuse of existing whole-part hierarchies or parts of hierarchies. This is also necessary because sometimes alternative synchronizations of the same basic parts have to be modeled as attachment-points for ICs. As an example, consider an audio recording for background music, where one track for each instrument exists. One hierarchy models all tracks together, a second hierarchy might model only certain tracks, e.g. the tracks with violins. The tracks of the second hierarchy can be played on their own or can serve as an anchor point for ICs. In this way, information concerning the violin-track only can also be stored.

single-valued/multi-valued: A whole can either have one part of a certain class *(single-valued)* or many parts of a certain class *(multi-valued)*.

Both the IOI-Extract and the IOI-Composite relationship are multi-valued.

range-restricted: A whole can have any number of parts of a certain class within a certain range.

Neither the IOI-Extract nor the IOI-Composite relationship is range-restricted.

fixed-cardinality: A whole has a specified number of parts of a certain class.

Neither the IOI-Extract nor the IOI-Composite relationship has a fixed cardinality.

ordered: An order is defined between the parts of a whole.

There is no order between the extracts of a certain IOI. We define all Temporal Composite relationships to be temporally ordered to ensure uniform treatment, even if this is not strictly necessary in the case of a Choose-Composite. Spatial Composite relationships are also ordered, the parts are placed, possibly overlapping, on the image to be created in the order defined by the relationships.

essential part: An essential part has a fixed cardinality of degree one.

IOIs do not have to have extracts because IOIs can exist whether they have extracts or not, many IOIs will never have extracts. Even though a composite has to have at least one part, there is no specific essential part.

dependency: The parts are destroyed if the whole they are derived from is destroyed.

Extracts are deleted when their IOI is deleted, because they are defined in terms of their IOI and do not have any meaning without them. An IOI is not deleted when a composite, which uses the IOI as a part, is deleted.

The Composite-IOI relationship is also dependent, because a Composite is meaningless without data. A composite is destroyed if its last part is deleted. If a part of a composite is deleted and at least one other part exists, the composite continues to exist, but can have a different duration or size after the deletion of its part. This can cause the need to update all wholes in the hierarchy above this composite.

value propagation: The values of specified attributes are propagated upwards or downwards the whole-part hierarchy.

Value propagation in terms of attributes of any kind of IOI can be handled as described in [Halper *et al.*, 1992]. More interesting is the propagation of the relationships between IOIs and ICs. These relationships are used to relate the abstract information modeled by the ICs to the concrete information of the IOIs. For example, as shown in Figure 5.6, a relationship occurrence called "directed by" relates an IOI connected to a video to an IC describing the person "Sue". The propagation of these relationships is described in the next section.

5.4.3.4 Relationship Propagation

The relationships between IOIs and ICs, shown in Figure 5.5 as Relates to, can be propagated upwards and/or downwards IOI-composition hierarchies. The following types of relationships between IOIs and ICs can be distinguished: base relationships, which are the relationships originally entered into the database, propagated relationships, which are derived from other relationships, and overloading relationships, which override a propagated relationship by changing the way the relationship is further propagated or by connecting the IOI to a different IC.

Base relationship: Base relationships are relationships between IOIs and ICs which have been inserted explicitly into the database. Base relationships can be propagated upwards and/or downwards or are not propagated at all. Base relationships are the only relationships which can be propagated upwards as well as downwards.

The propagation mode describes in which direction a relationship is propagated and how a relationship is propagated. If only the given relationship is propagated, the propagation is called *simple*. For example, in most cases the person directing the video will have directed all parts of the video. This can be expressed by defining the "directed by" relationship occurrence of the top composite as downward and simple. The other kind of propagation, aggregated propagation, is discussed below.

In Figure 5.6, the propagation mode is shown in parenthesis after the name of the relationship occurrence: the first letter is either "u" for up or "d" for down, the second letter is "s" for single or "a" for aggregated (see below).

Propagated relationship: A propagated relationship exists because a base relationship is propagated in the direction of a given IOI along the whole-part hierarchy. For example, in Figure 5.6 the relationship occurrences "filmed by" are propagated upwards to IOI "Video Linz 1".

Propagated relationships receive the propagation mode which is relevant in their direction, i.e. a propagated relationship below the base relationship

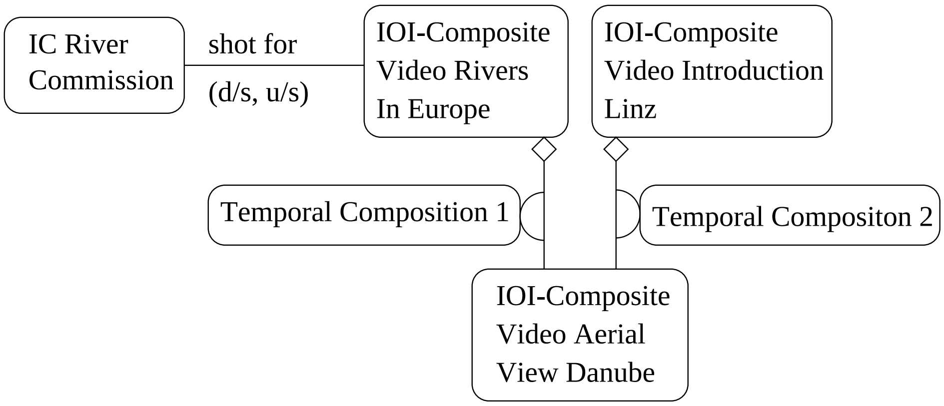

Figure 5.11: "Yo-yo Effect"

only has the information how the relationship is propagated downwards. This prohibits a "yo-yo-effect", as shown in Figure 5.11: The relationship occurrence between IC "River Commission" and Composite "Video Rivers in Europe" is defined as down/simple and up/simple. Without the mechanism described above, the propagated relationship occurrence between IC "River Commission" and Composite "Video Aerial View Danube" would also be down/simple and up/simple. From Composite "Video Aerial View Danube" the relationship occurrence would be propagated to Composite "Video Introduction Linz" and all IOIs connected to Composite "Video Introduction Linz".

A propagated relationship can redefine the relationship into *aggregate,* i.e. all like-named relationships which are propagated to this IOI are now collected into one set. This set contains the union of all like-named relationships, possibly including previously aggregated sets. The result is always a set of relationship occurrences, but never a set of sets. The direction of the propagation may not be changed. The resulting set is propagated further as simple, i.e. the aggregate-property has to be set at every IOI where relationships should be combined into a set. If two different relationships of the same name are propagated to the same IOI without being aggregated, an error is raised, regardless whether the propagated relationships are simple or aggregated.

The camera person can change at each shot, so the relationship occurrence "filmed by" between e.g. "IOI 3" and "IC Jeff" is upward/simple. At the first IOI which combines the work of several camera persons, in Figure 5.6 "Video Linz 1", the propagation is changed to aggregated, thus resulting in a set of camera persons (not shown in Figure 5.6).

Overloading relationship: Overloading relationships overload the propagated relationship of the same name by changing the way the relationship is propagated further or by connecting the IOI to a different IC. For example, "IOI 3" is scripted by Sue, while all other IOIs are scripted by Joe.

The overloading is only effective in one direction, e.g. if the overloading relationship is below the base relationship, the overloading relationship is propagated downwards only. This restriction is necessary because otherwise inconsistencies between the overloaded and the base relationships or between overloaded relationships can occur.

Optionally the propagation of relationships can be restricted to whole-part relationships within the same type of relationship, e.g., the propagation can be restricted to IOI-Extract relationships.

In some situations the ability to overload base relationships arbitrarily can be undesirable, because the creator of the relationship has no control on how the relationship is overloaded. Also for users of propagated relationships it is sometimes necessary to know that the relationship will not change due to the introduction of an overloading relationship. For these reasons the distinction between "strong" and "weak" base relationships is introduced: only weak base relationships can be overloaded by overloading relationships. For example, in Figure 5.6 the relationship occurrence "directed by" between "IC Sue" and "Video Linz 1" is strong, while the relationship occurrence "scripted by" between "IC Joe" and "Video Linz 1" is weak and is overloaded for "IOI 3" by the relationship occurrence with "IC Sue". Note: whether a relationship occurrence is weak or strong is not indicated in Figure 5.6.

5.4.3.5 Whole-Part Relationships in existing Hypermedia Data Models and Hypermedia DBMS

Whole-part relationships are used in a variety of ways in hypermedia modeling. One commonly used type is the composite, which is a collection of nodes. For a survey of the concept of composites, especially for older systems, see [Grønbæk, 1994], which also describes the implementation of composites in the context of the Dexter Hypertext Reference Model [Halasz and Schwartz, 1994]. This section discusses selected research results to show how whole-part relationships are used in hypermedia modeling.

In [Bra and Houben, 1992] composite object constructors are used to create composite objects, which can consist of links, anchors, and nodes. No semantic interpretation is given, although it is mentioned that this mechanism can be used to create, for example, time-lines and maps.

Composites with built-in semantics, i.e. composites which have some meaning in addition to the whole-part meaning, for example temporal composition, are described in [Garzotto *et al.*, 1994], where a collection can impose a structure upon its nodes, for example sequence, tree, or guided tours, and in [Maurer *et al.*, 1993b], where collections structure the nodes in envelopes (each node is connected to every other node), folders (sequences of nodes), maps, and menus (hierarchies of nodes).

Hierarchical composites are a common abstraction used in modeling hypertext. They are familiar, because "classical" media like books are also structured hierarchically. Botafogo et. al. [Botafogo *et al.*, 1992] describe a mechanism to extract the hierarchies of a given (unstructured) hypertext automatically and discuss visualisation techniques like maps, outlines, and the fisheye lens model.

In HDM [Garzotto *et al.*, 1993d] hierarchies are used as a purely syntactic device to organize information into manageable blocks within a possibly larger application. In HDM these hierarchies do not have any semantic meaning, as HDM is not intended to be a semantic model.

Another typical use of whole-part relationships are aggregation hierarchies, which play a central role in the modeling of multimedia documents in [Woelk *et al.*, 1986]. They are used to model the structure of a document, which can also be used for simple sequential temporal modeling and—using a different hierarchy—for modeling the physical layout on a page. Nodes of the aggregation can have properties, which can be propagated downwards, e.g. font size. Instantiated wholes can have additional parts.

In [Schnase *et al.*, 1993] whole-part relationships are used as type constructors in the Saberel semantic data model. The relationship does not have any additional meaning, i.e. it is left to the application whether the whole is a new object or simply a collection of objects. It is also not defined how the parts are related. Aggregations are used in a similar manner in the formal hypertext model described in [Garg, 1988].

Whole-part relationships with a particular meaning are used less frequently. In [Mayfield and Nicholas, 1993] aggregation is not used to model how multimedia nodes are composed, but to show how the information contained in hypertext nodes is related. While this approach can be compared to our modeling of information using IC-Objects, the approach presented in [Mayfield and Nicholas, 1993] is far less powerful and restricted to textual data. See also the discussion in Section 5.4.2.

5.5 Presentation Level

In this section we describe the presentation level of the HyperD architecture. The presentation schema describes how the information given in the content schema is made available to the user in the context of a specific application. The goal of the presentation schema is to model *which* information as modeled in the content schema is to be made available to the application and thus to the user and to model *how* the information is to be connected. This includes the decision which links and thus which information is available to the user as well as entry points and guided tours through the information. If desired, the classes in the presentation level can be extended to store the information needed to control the actual presentation of the hypermedia data on screen. This information includes for example the position of an image on the screen or the volume at which an audio track is to be played. Yet because this information is highly dependent on the application using the database, it is not defined in the architecture.

The presentation schema comprises all views defined for a hypermedia database. It is separated from the content schema in order to provide logical data independence. We first describe the purpose of the presentation schema. Then we describe the object model of the presentation schema and discuss how application specific knowledge is modeled. The section thereafter describes the meta model of the HyperD architecture, which also supports the display of the classes in the network of links and nodes presented to the user.

In this thesis we use the term “view” as the set of all objects visible to a certain user or a group of users of a database. All views of a database taken together constitute the presentation level of the database.

Hypermedia links, for a discussion of the term see Section 2.1.4.2, are an important modeling concept in the HyperD architecture. While in the content level the term “relationship” is used to emphasize the data modeling aspect, in the presentation level the term “link” is used instead to emphasize the navigational aspect. As discussed later, one important difference is that a relationship in the content level is bidirectional, while a link in the presentation level is unidirectional.

5.5.1 Purpose of the Presentation Level

In the presentation level the information described in the content schema is rearranged to reflect the needs of the users of the database. The users are thus insulated from most changes occurring in the content schema. Another benefit of logical data independence is the separation of “general knowledge”, which is of interest for many applications, from "application knowledge", which is of interest for one only application. General knowledge is modeled in the content schema, so that all applications can benefit, and application knowledge is modeled in the external schemata, so that the content schema is not cluttered unnecessarily.

A view reduces the huge amount of data present in a database to a smaller amount of data actually needed by some users. This can be done by (a) restricting the number of objects a user can see or (b) by aggregating information. Typical commands used to create view-relations in relational databases are joins, projections, and selections. They are applied to make only certain data available to a certain user, e.g. data created by him or her. The analogue task in a hypermedia database is the selection of the media objects a user should see by combining the content of certain nodes and by restricting the number of links available.

The differences of the approach of views as implemented in the HyperD architecture to the various approaches as implemented in object oriented database management systems are discussed in Section 6.3.

The decision which information to make available can be made at the class level, e.g. to include a certain relationship for all instances of a class, or for a specific instance. The selection can either be made explicitly by enumerating links or by selecting links based on queries. Queries can use certain properties of links, e.g. class, owner, security clearance, or value of attributes, to select links.

The following operations reduce the number of links available to the user:

- Selection of one component of a Choose-composite (cf. Section 5.4.3.1), e.g. selecting the English commentary.
- Selecting relationships between Conceptual Items to be included in the view.

The following operations add links to the database.

- Adding of application links (i.e. links which are only of interest to a specific application).

- Adding of predefined paths (guided tours) through the link and node network.
- Adding of "aggregated" links, i.e. links that result from the combination of more than one link of the content schema.

The object model described in the next section allows the implementation of these operations.

5.5.2 Object Model of the Presentation Schema

The basic idea of our approach is simple: every Conceptual Item in the content schema can be included in a view by creating a Presentation Item connected to the Conceptual Item. Presentation Items among themselves are connected by Presentation Links, which correspond to one or more Conceptual Relationships. Yet, to implement this view mechanism one obstacle has to be tackled: relationships in the content schema are bidirectional. Since in views we want to traverse links always in one direction using a query language, we model only uni-directional links (or sets of links) in the presentation schema. This consideration leads to the inclusion of class Link Definition, where each Link Definition is related to one or more presentation items. The set of presentation items associated with a Link Definition is either computed once upon creation of the link definition (Static Link Definition) or each time the link definition is accessed (Dynamic Link Definition). Class Presentation Link can be subclassed similar to conceptual relationships to provide object oriented modeling. For presentation links, which do not belong to an user-defined class, class Generic Presentation Link (shown in Figure 5.12 as Generic PL) has to be used. Figure 5.12 shows the object model of the presentation schema. At the bottom of Figure 5.12 classes Conceptual Item and Conceptual Relationship of the content schema are shown again.

Each view is represented by an object of class View, to which certain presen-

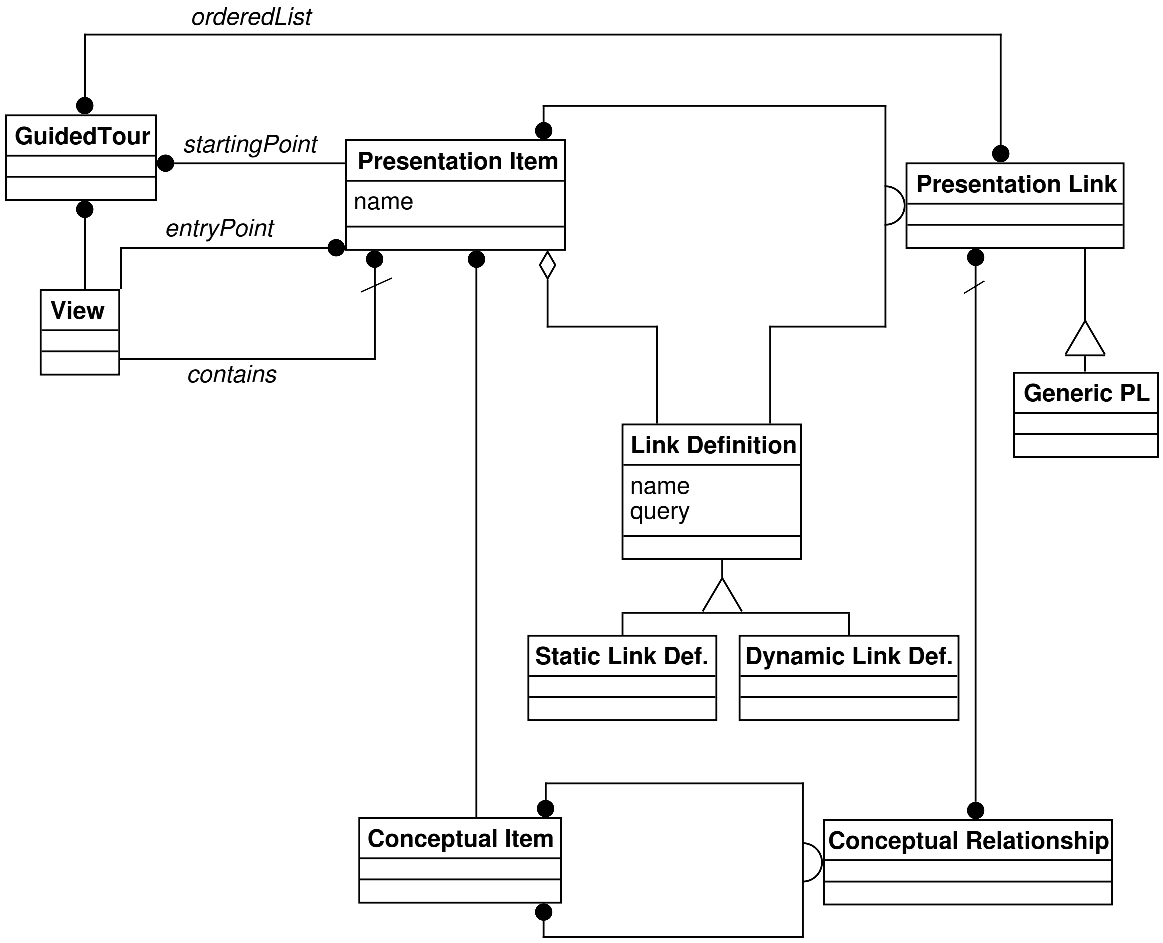

Figure 5.12: Object Model of the Presentation Schema

tation items are connected with an entryPoint relationship. These presentation items serve as entry points to networks of presentation items. All presentation items which can be reached from these entry points also belong to the view and are connected to the view with the derived contains relationship.

5.5.2.1 Example

As an example, Figure 5.13 shows an extract of a database modeling sights in a city, in this case Linz. The sights of Linz are shown in the content schema as instances of class Sights, which is a subclass of IC, and are connected by the IC-Relationship "is-near", e.g. the "Nibelungenbruecke" is-near the "Altes Rathaus". In the media schema, two videos of local sights are shown. The classes defined in the HyperD architecture are shown as rectangles with thick lines, all other classes are subclasses of one of the predefined classes.

Note: class Guided Tour refers to a special construct used in hypermedia modeling and is described in Section 5.5.4.1; class Tour, which is a subclass of Guided Tour, models an actual tour through the city. The relationship occurrences between the relationship occurrences in the content schema and presentation links are not shown in Figure 5.13, instances of Temporal Composition are not shown either.

5.5.2.2 Presentation Items

A Presentation Item represents a Conceptual Item in the presentation schema. For each subclass of Logical Medium, a subclass of Presentation Item exists (see Figure 5.14). Presentation items may only be connected to matching IOIs, i.e. a Video Presentation Item may only be connected to an IOI representing a Logical Video and a Composite Presentation Item may only be connected to a Composite. An Extract has the same media type as the IOI it is derived from, i.e. an extract is represented by a Composite Presentation Item if it has been derived from a Composite or it is represented by an Audio Presentation Item if it was derived

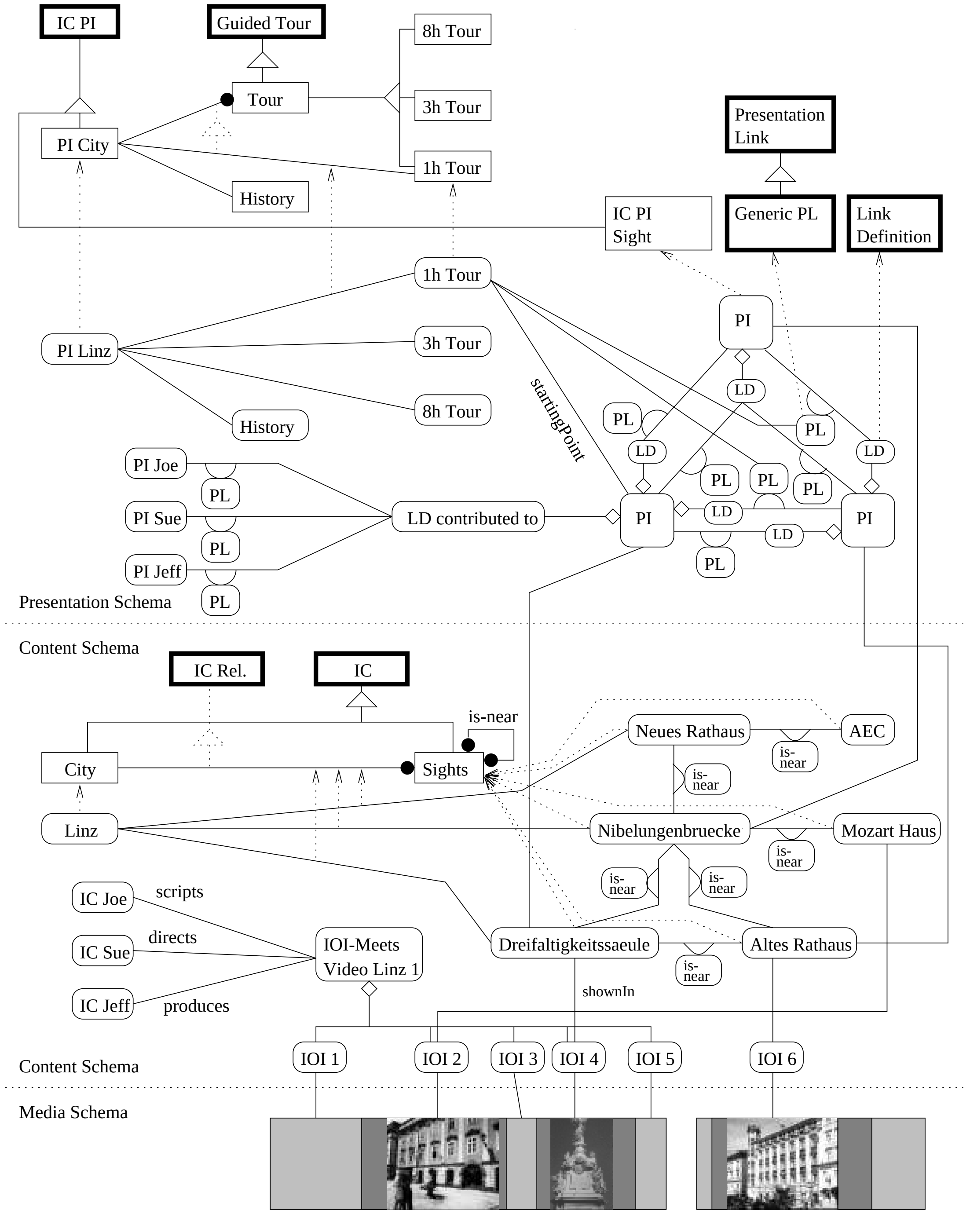

Figure 5.13: Example Presentation Schema

from a Basic IOI connected to a Logical Audio. Composite Presentation Item is a subclass of all subclasses of Presentation Item, except IC-Presentation Item, because a composed conceptual item can contain any kind of IOI.

As discussed above, it is straightforward to derive subclasses from the core presentation item classes, which contain the information needed for the presentation on screen. Since those attributes are highly application specific they have not been made part of the HyperD architecture.

One advantage of this approach is the seamless integration of conventional data, represented through attributes of IC-Objects, and hypermedia data. IC-Objects, represented in the presentation schema by class IC Presentation Item are shown on the screen generically, e.g. in the familiar OMT notation, and can be used like any other presentation item. Another advantage is that if a presentation item is reconnected to a different conceptual item (which has to be compatible, i.e. another image), the presentation description is automatically applied to the new image, which is integrated seamlessly. Subclasses of IC Presentation Item can be created to model concepts. For example, one subclass can be City, as shown in Figure 5.13. Each city, e.g. Linz, is described by general information and some sightseeing tours. Class Generic IC Presentation Item has to be used for IC Presentation Items for which no user defined class has been created, because otherwise the rule, that only leaf classes of inheritance hierarchies can have instances would be violated.

There is exactly one Conceptual Item for a Presentation Item, i.e. we do not allow the aggregation of several Conceptual Items into one Presentation Item. Aggregations of conceptual items have to be performed in the content schema.

5.5.2.3 Link Definitions and Presentation Links

While there is always a one to one mapping between presentation items and conceptual items, the same is not true for presentation links and conceptual relationships.

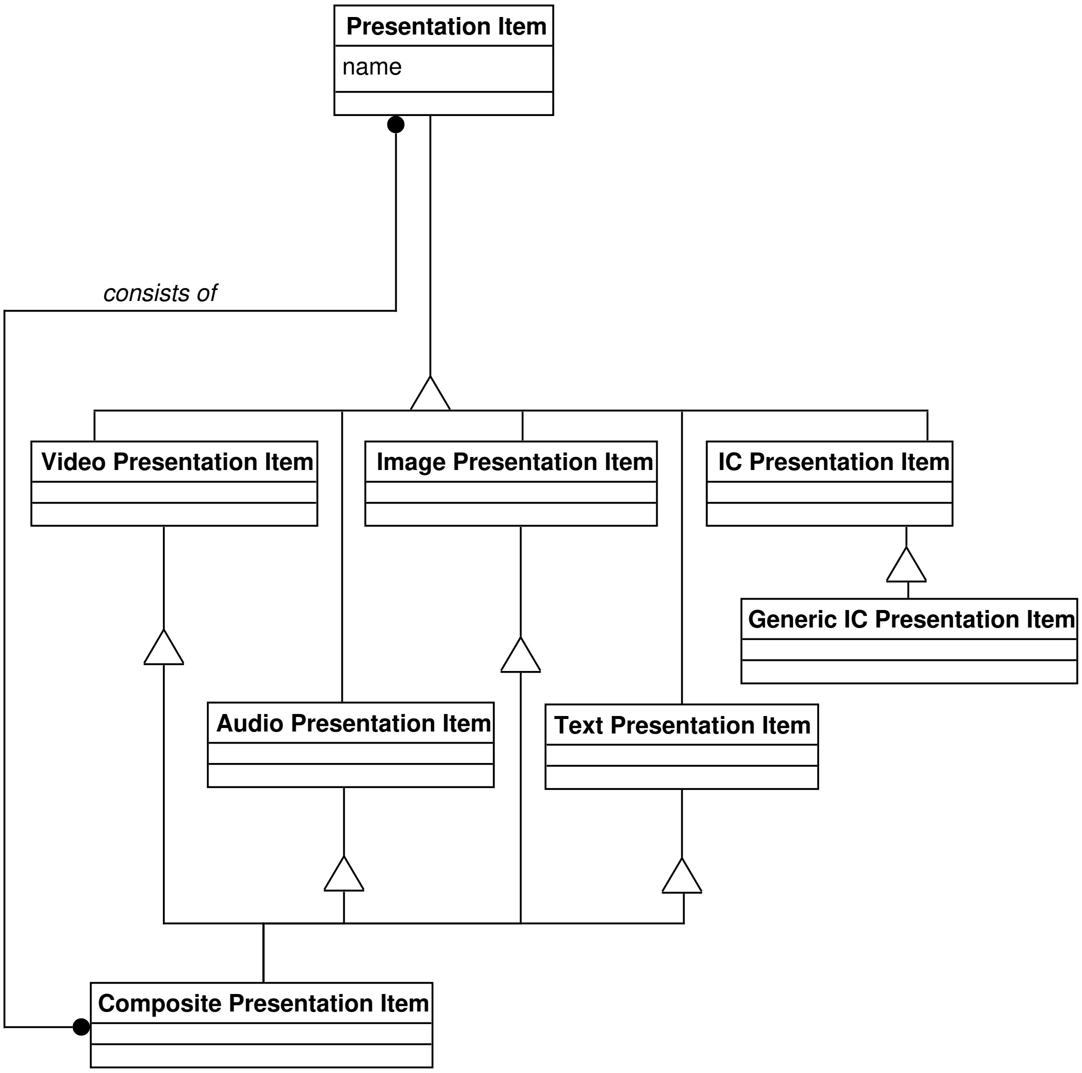

Figure 5.14: Detail from the Object Model of the Content Schema: Presentation Items

As stated above, conceptual relationships are bidirectional. Links in the presentation schema are used to traverse from one presentation item to another presentation item, i.e. they are used unidirectional. To achieve a high degree of decoupling between the content level and the presentation level, queries are used to select the relationship occurrences to be included in the presentation schema and thus to select which links are available to traverse from a certain presentation item. Therefore, one bidirectional relationship occurrence in the content schema is modeled by two unidirectional links in the presentation schema. A different query has to be stored for each of these two links because the queries start at different presentation items and use the path between these items in different directions. Each query can evaluate into a set of links. A very simple query language is described below.

Queries are stored in objects of class Link Definition. Two classes of link definitions can be distinguished:

- Static Link Definition evaluates the associated query upon creation and has the set of associated presentation links stored explicitly.
- Dynamic Link Definition evaluates the associated query to compute the set of presentation links each time the Dynamic Link Definition is accessed. If a dynamic link definition object is dereferenced, the query is evaluated and the appropriate presentation links, i.e. instances of class Presentation Link are generated. Each of these links has a derived relationship occurrence (shown in Figure 5.12 with an additional diagonal line) to the conceptual relationships it corresponds to.

  The presentation links are deleted at the end of the transaction in which they were created. If the same Dynamic Link Definition is accessed more than once in the same transaction, the query is re-evaluated, because the content schema could have changed. Existing presentation links are re-used if possible.

The existence of a Link Definition depends upon the existence of a corresponding Presentation Item, hence the aggregation relationship between Link Definition and

Presentation Item.

The combination of link definitions and presentation links is necessary only to model a set of links, a presentation link would be sufficient to model a single link. To treat links uniformly and to simplify the object model we have chosen not to include special presentation links representing a single conceptual relationship and to require a combination of a link definition and a presentation link also for single links. A Link Definition must be created for each link.

Every Conceptual Relationship can be used in the query expression, thus every conceptual item can be connected to every other conceptual item in a view without the need for a direct connection. Although these presentation links can make arbitrary connections, they only contain the application specific knowledge which nodes to connect, but they do not add new knowledge in the form of *new* relationships between conceptual items.

Class Link Definition can be subclassed to provide particular behavior, e.g. presenting the links in a certain order or presenting only the three most relevant links. Subclasses of Link Definition can also specify how the presentation item is to be displayed, e.g. if it replaces the current item when the link is traversed, if it is inserted into the current item or if it is displayed in a new window.

In the simplest case, the query consists of the name of a conceptual relationship, thus representing the conceptual relationship in the presentation schema. More complex queries can use sequences of conceptual relationships and restrictions based on attributes of conceptual items. The query language used depends on the database management system chosen for implementing the architecture.

A simple query language could consist of:

- Path expressions in the form of "relationshipname"."relationshipname". For example, the expression "shownIn.scripts" leads from "Dreifaltigkeitssaeule" to "IC Joe".

- Set operators "∪" and "∩", which can be used to manipulate sets of relationship occurrences.

For example, the link "contributed to" evaluates to three relationship occurrences in Figure 5.13: "scripts", "directs", and "produces". As can be seen in Figure 5.6, the relationship occurrences "scrips", "directs", and "produces" are propagated to "IOI 4", which is represented by "Dreifaltigkeitssaeule", which in turn is represented by the presentation item in the presentation schema. The query which led to the link "contributed to" is "shownIn.scripts ∪ shownIn.directs ∪ shownIn.produces".

As this thesis describes a general architecture, we do not describe the query language in greater detail. Possible query languages include HyQ [Kimber, 1993], HyOQL [Gardarin and Yoon, 1994], and O_2SQL [Christophides and Rizk, 1994].

There are also some specialized query languages which can be adapted for the HyperD architecture. One example is StruQL [Fernandez *et al.*, 1997], which is intended to be a query language for web-site management systems. StruQL is the query language of the STRUDEL system, which uses a graph based data model consisting of nodes and labeled links. The data model provides an integrated view of internal and external resources. StruQL can be used to query or transform graphs. STRUDEL implements a pure link and node model and could be extended by the constructs and classes as described in this thesis.

Another possible query language is Lorel, which is the query language of the Lore system [McHugh *et al.*, 1997]. Lore is a database management system for semistructured data. Lorel is an OQL extension and is implemented as a precompiler, which compiles Lorel expressions into OQL expressions. Lorel offers general path expressions, which may contain set valued attributes at any point (OQL allows set valued attributes only as the last attribute of a path expression). Lore uses the OEM (Object Exchange Model) as the data model. A lore database consists of a graph of atomic objects, complex objects, and labeled edges. Atomic objects contain data from basic atomic types such as integer and string and are not allowed to have outgoing edges. Complex objects have outgoing edges, but cannot have data. A OEM database does not have a fixed schema, all schematic information is expressed by the labels of the edges, which may change dynamically.

A Presentation Link connects two or more presentation items. The following cases show the possible relationships between presentation links and conceptual relationships:

- A presentation link can represent a single conceptual relationship.
- A presentation link can represent a series of conceptual relationships, e.g. the presentation link "Jeff works-with Joe" (not shown in Figure 5.13) can represent the conceptual relationship occurrences "Jeff produces Video Linz 1" and "Video Linz 1 is-scripted-by Joe".
- A presentation link can represent a set of conceptual relationship, as shown for example in Figure 5.13: the content schema contains relationship occurrences stating that Joe scripted, Sue directed, and Jeff produced the video containing the segment about the Dreifaltigkeitssaeule (Three relationship are propagated downwards from the IOI representing the whole video to all segments of the video). The relationship occurrences "scripts", "directs", and "produces" are combined into the link definition "contributed to" in the presentation schema.

Combinations of the last two cases are also possible. In addition, select statements can be used to filter out certain conceptual relationships from a set.

5.5.3 Meta Schema

One design goal of the HyperD architecture has been the ability to represent selected classes in a view so that not only instances of a subclass of Presentation Item can be displayed, but also the classes the instances belong to. For this reason, the object models of the presentation schema and the content schema are

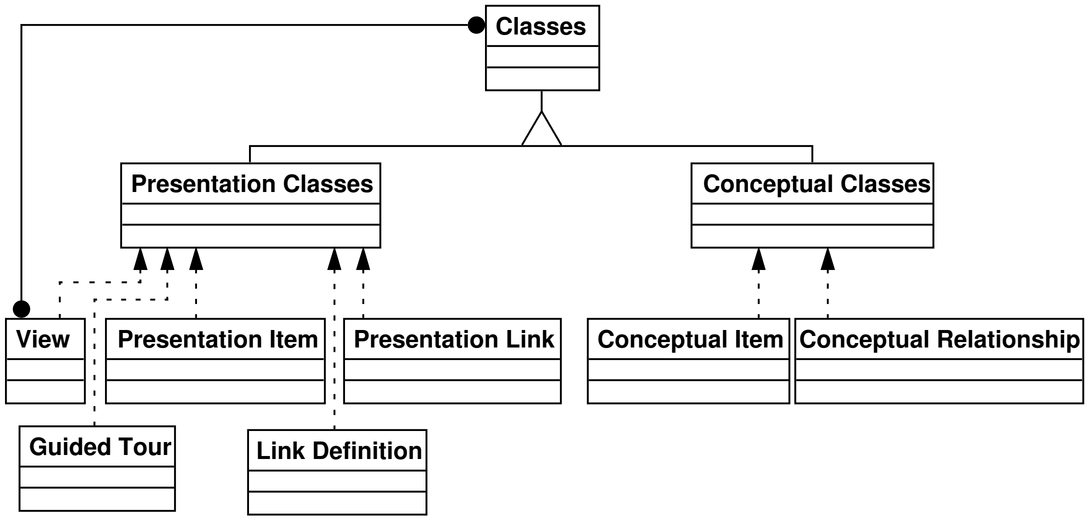

Figure 5.15: Object Model of the Meta Schema

extended as shown in Figure 5.15. All classes of the presentation schema, i.e. Presentation Item, Presentation Link, Link Definition, View, and Guided Tour and also their subclasses are instances of meta class Presentation Classes, which in turn is a subclass of meta class Classes. In Figure 5.15, the instance-of relationships are shown as dotted arrows. Instances of Classes can be associated with views, and thus be made visible to the user of the view. For example, in Figure 5.13 the user can see both class City and its relationship to Tour and History as well as an instance of City, Linz, and instances of class Tour.

Conceptual Item and Conceptual Relationship are defined as instances of another subclass of Classes, Conceptual Classes.

5.5.4 Application Specific Knowledge

Application specific knowledge is of interest to only one application. While in most cases views are used to hide information, in conventional databases they can also be used to add aggregated or otherwise calculated information. The analogue design decision in hypermedia databases is which information can be represented in the presentation schema without being also stored in the content schema. To be able to treat all media data uniformly in the HyperD architecture, all objects associated with media data have to be stored in the content schema. Additional links are allowed in views for the following reasons:

- Including every link as a relationship in the content schema would add details to the content schema which are application specific and thus are not relevant for other applications.
- If an additional link is needed in an application, the content schema would have to be changed. Such changes cannot be made by the application designer because these changes require the privileges of the database administrator.

Another kind of application specific knowledge is modeled by additional objects in the presentation schema, e.g. by instances of Guided Tour.

The following sections discuss some constructs commonly found in hypermedia/hypertext data modeling, which are examples for application specific knowledge.

5.5.4.1 Guided Tours

Guided tours are predefined paths through the network of links and nodes and can be navigated simply by giving the command to visit the next or previous node. They are intended as a means to help a novice user to find her way.

Guided tours are defined upon existing links, thus the only "new" information provided by the guided tour is the sequence in which links are followed. We consider this knowledge to be application specific, thus guided tours are modeled in the presentation schema: each Guided Tour consists of a starting point (the first node to be displayed) and an ordered sequence of links.

As the links used in a guided tour are presentation links, they do not have to exist explicitly in the content schema, it is sufficient if they can be derived from the content schema. For example, no previous/next relationships are necessary in the content schema: for a guided tour showing all persons who contributed to the video, the link from Jeff to Joe can be derived from the conceptual relationships “Jeff produces Video Linz 1” and “Video Linz 1 is-scripted-by Joe” (not shown in Figure 5.13).

5.5.4.2 Table of Contents/Indices

Another construct often found in hypermedia applications are nodes which serve as a table of contents or as indices. Often these nodes cannot be simply derived from the links, but have some artwork or formatting associated with them. This artwork constitutes a media object with associated media data and such media objects are only allowed in the content schema. For this reason it necessary to include these media objects in the content schema.

The links between these table-of-contents or index nodes and other nodes add no information of general interest to the database and are thus classified as application specific, so the links to and from these nodes only need to be present in the presentation schema.

Chapter 6

Data Independence in HyperD

This chapter discusses how data independence is achieved in the HyperD architecture. Every level imposes new costs, e.g. because of the mapping operations, which have to be justified by some advantages, especially some kind of data independence. These advantages are described in this chapter.

The HyperD architecture is shown in Figure 6.1, where it is contrasted to the ANSI/SPARC architecture. The kind of data independence achieved by separating the information in two adjacent levels is shown next to the levels.

In relational databases physical data independence is responsible for insulating the conceptual schema from the concerns of the physical storage of the entities represented in the conceptual schema. Typical considerations are the clustering or fragmentation of relations, the instantiation of additional access paths, the decision if derived attributes are stored redundantly or computed when needed, or simply the storage organization used for relations. For a discussion of data independence in relational database management systems see Chapter 4.

Applying physical data independence to hypermedia database management systems can be decomposed into two distinct kinds of independence: the media level is concerned with the selection of the appropriate media data and the sto-

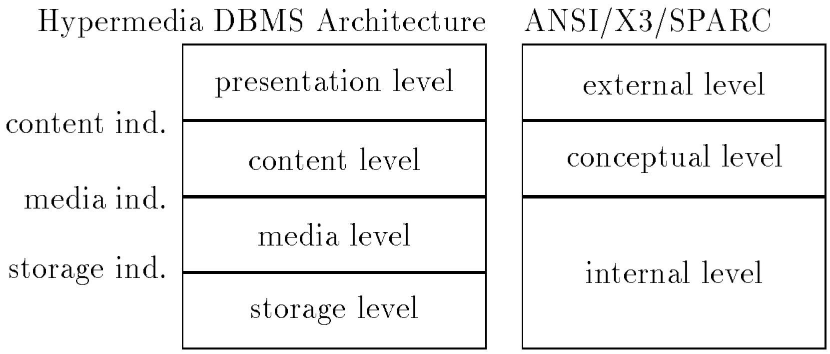

Figure 6.1: DBMS Architectures

rage level with the physical storage of the artifact. This distinction is comparable to distribution transparency in distributed databases, which is further differentiated into fragmentation and allocation transparency. Storage independence and media independence together play a similar role as physical data independence in relational database management systems.

The concept of logical data independence can be mapped straightforward to the concept of content independence as used in the HyperD architecture. The term "content independence" is used instead of the term "logical data independence" to allow for a consistent naming of the various kinds of independencies in the HyperD architecture. Content independence is described in Section 6.3.

6.1 Storage Independence

Storage independence is achieved by separating the information described in the media schema from the information described in the storage schema. Storage independence hides the storage details of media artefacts, e.g. format, resolution, and segmentation, from the applications using the data. It helps to achieve the following goals:

- *Easier reuse of multimedia objects:* Often different applications use the same multimedia object, but each application requires a slightly different variation of the object. Variations needed include, e.g. different segments of the object, different encoding, and transformations (for images, e.g. gamma correction and effect filter). Without data independence, these variations are stored separately, thus creating uncontrolled redundancy. Alternatively variations may be created on demand, based on specifications supplied by the user. A mechanism is needed to provide these variations on demand in a consistent and efficient manner.

  As mentioned in Section 5.2.2, we have chosen the term "variation" to contrast it to "derivation" as used e.g. in [Gibbs *et al.*, 1994], which is used to describe a view mechanism *and* a mechanism to achieve physical data independence.

- *Higher performance:* Operations needed to transform the multimedia objects often last too long for interactive presentations. To achieve higher performance, the need of storing different variations remains in order to avoid time consuming transformations. A mechanism is needed to instantiate and control the use of customized variations of a multimedia object.

For a discussion of how these goals are achieved see also Section 5.3.4.

Storage independence has an objective similar to the objective of the conceptual-to-physical mapping in relational databases: it hides the decisions made about the physical storage of the media object. The logical representation of a media object describes a medium with certain characteristics, e.g. length for an audio or video track and size for an image. It does not show:

- how the medium is stored, e.g. if a video is a video or a sequence of pictures and which method was used to encode the video
- where the medium is stored
- on which platform the medium is stored (laserdisk or harddisk)

- whether the medium is stored as one chunk or whether it is composed of several chunks
- whether the same medium exists in different variations (resolution, frames per second) to save processing power at replay-time

The advantages are similar to those described for the logical-to-physical mapping in relational databases: decisions made to save space or decisions made to accelerate certain queries by allowing redundancy as well as decisions about the location of a media object do not necessitate changes in the logical level.

For an example, especially for the performance gains possible, see Section 5.2.3.

6.2 Media Independence

Media independence insulates applications accessing some (abstract) information from changes to the relationship between the information and the media data representing this information. Media independence is achieved by separating the information contained in the content schema from the information described in the media schema.

This insulation is achieved by mapping the abstract information about a fact of the real world in the content schema to a part or a span of multimedia data contained in the media schema: One IOI could be a for example "Entrance to MZ" and is connected to a part of an image showing the building. The part of the media object the IOI is connected to can overlap with the part used by a different IOI, e.g. "Students" (see Figure 5.3).

Media independence solves the problem of semantic content linking integrity as discussed in [Campbell and Chung, 1995]: Media objects can be used for more than one purpose, the image mentioned above can be used for the IOI "Front

View Joe" and for the IOI "Entrance to MZ". If the image is simply replaced with a newly shot image showing Joe with his new glasses, the semantic content linking integrity for the second IOI is violated, because the new image does not show the entrance to the Management Zentrum. Since, as mentioned in [Prückler and Schrefl, 1995], the extents of media data referenced by IOIs can overlap, IOIs can be used to determine which parts of a media object are referenced, which shows for which purposes media objects are used. Thus IOIs can also be used to safely introduce new objects without disturbing the usage of already existing objects.

The following example shows how media independence is achieved: Joe gets different glasses, thus a new image is shot for the IOI "Front View Joe". The only change necessary is to connect the IOI to the new image—no other IOIs are changed. This insulation also works if the image is replaced by a video.

6.3 Content Independence

Content independence is achieved by separating the information described by the content schema from the information described by the presentation schema. The goal of content independence is the same as the goal of logical data independence. The term "content independence" is used instead of "logical data independence" to achieve consistent naming of the independencies in the HyperD architecture.

The problem of how to achieve logical data independence is similar for object-oriented database management systems and hypermedia database management systems. First, because many hypermedia database management systems are implemented using object-oriented database management systems and second, because objects and object references are represented in a similar manner as hypermedia data and their links, by graphs. Usually, views are used to achieve logical data independence by providing an additional layer between the conceptual description of the data and the application using the data. The question of how to support views, especially in a manner which allows consistent updates, is an area of ongoing research in the area of object-oriented database management systems and only few object-oriented database management systems contain constructs to create views [Dobrovnik, 1995]. One example are object-preserving views which allow to propagate updates to base objects [Scholl *et al.*, 1991].

Yet there are some aspects in which hypermedia data differ from conventional data, as modeled in relational or object-oriented databases:

- One important issue for views in object oriented database management systems is the following question: the objects, which are the results of the queries on which views are based, are instances of which class? This is especially difficult if an object is the result of a join of more than one base objects, because then a new class has to be created and the class has to be inserted into the inheritance hierarchy. Another related question is whether new objects are created in views or are just interfaces to existing objects. This also affects how updates can be handled.

  These questions are no issues in the context of the HyperD architecture, because objects can only be instances of a predefined class. Thus using the approaches applied in object oriented database management systems would be unnecessarily complicated and thus would not be economical.

- Views in the HyperD architecture can be defined at the class level, as it is done in views in object oriented database management systems, but also at the level of instances, i.e. single objects of the content schema can be part of a view without the need to include their classes. This is especially beneficial for mapping into simple link and node models without higher semantics. For example, as shown in Figure 5.13, whole classes (e.g. class "PI City") can be modeled in the presentation schema, but also single instances (e.g. "PI Sue" or "PI Jeff") which do not belong to a user defined class. Note: in Figure 5.13 the instance-of relationships between "PI Sue", "PI Jeff", and "PI Joe" and class Generic Presentation Item are not shown.

In relational databases logical data independence is achieved by "hiding" base relations behind view-relations, so that when base relations change only the definitions of the view-relations have to be modified. In many cases the external appearance of view-relations does not change, so applications using the view need not be modified. Presentation items and presentation links are used to "hide" changes of the content schema, thus presenting a structurally unchanged network of presentation items and presentation links to the applications using the database:

Presentation Items can be "redirected" to other conceptual items thus showing e.g. a different image using the same presentation parameters. For example, if an application shows the image of the producer of a video in the bottom left corner and the producer changes, the image showing the new producer is used automatically in the same place scaled to the same size as the picture of the previous producer.

Changes to the media data are hidden from presentation items by conceptual items: if an application shows the image of the producer and a new image of the producer is inserted into the database, the new image is used automatically.

Link Definitions can hide changes by simply replacing the query expression, thus changes to the conceptual structure do not have to be propagated into the applications using the data.

Changes to the content of the data can be propagated to the views automatically. For example, one query can return a link to the city with the lowest crime rate. If the crime rate changes, the link is automatically redirected to the correct city.

One other important benefit of this approach is that it works both at the instance level, where particular objects of the content schema are selected disregarding the class they are an instance of, and at the class level, where classes and their instances are selected for inclusion in a view. This gives the user full flexibility to model the data either as (a) a simple, unstructured network of links and nodes, (b) as a sophisticated semantic network, or (c) a mix of both approaches.

Chapter 7

Corresponding Information Content

This chapter describes an extension of the **Hyper**media **D**atabase Management System Architecture. By extending the HyperD architecture with Corresponding Information Content (CIC) a higher degree of expressiveness when modeling hypermedia data can be achieved.

We have chosen to present the concept of CIC in a separate chapter for the following reasons:

- Corresponding Information Content is an extension to the HyperD architecture, in that the HyperD architecture is also useful without CIC. Therefore, discussing CIC in a separate chapter helps to achieve a better presentation of the "core" architecture by not distracting attention to an optional extension.
- We want to emphasize that this is an extension to the original architecture and thus shows that the HyperD architecture can be extended easily.

The results presented in this chapter were first published in [Prückler and

Schrefl, 1995]. The term "piece of information" is used in this chapter instead of the term "item of information" as used in [Prückler and Schrefl, 1995], because in the context of the HyperD architecture the term "item of information" is used differently.

The Section 7.1 discusses the concept of Corresponding Information Content. Section 7.2 describes the properties of CIC-relationships and Section 7.3 describes how the concept of CIC-relationships can be incorporated into the HyperD architecture.

Throughout the chapter the example shown in Figure 7.1 will be used. The objects shown in the figure model information about the Dreifaltigkeitssäule (the Pillar of the Holy Trinity), which is a sight in Linz. The example is explained in Section 7.2.

7.1 Basic concept

This section discusses the motivation and the basic concept of Corresponding Information Content.

7.1.1 Nature of Hypermedia Data

The nature of hypermedia data differs substantially from "classical" alphanumerical data. Some notable differences in the context of this work are:

- One attribute of structured data contains exactly one piece of information, for example the attribute `COMPLETED` contains the value "15. 5. 1723" bearing the information that the pillar was completed on May 15, 1723. One piece of multimedia data can convey many different pieces of information, e.g. a video-recording of a guided tour through the city contains the date of the completion and other information about the pillar.

- One piece of information exists exactly once in a relational database (disregarding access structures), while one piece of information can be contained in many different media objects. The information that the pillar was completed in 1723 can be contained in a tourism folder and in a video-recording of a city tour.

The term “piece of information” means the following in the context of this chapter: Data is not stored without reason, but to convey a certain meaning. An organizational chart is hardly ever stored because it is a pretty picture but because it expresses the organizational structure of a company. Data is stored for a specific reason, for example “Because I need it later” or “I want to show the user something”. To put it another way, the basic rationale is to conserve some meaning, which can be used later. The basic concept which represents this intention is the Piece of Information: A *Piece of Information* is the meaning the creator of a media object wanted to convey by the media object or by a part of the media object when he created the object or the links to the object.

7.1.2 Redundancy in Hypermedia Data

As shown in the previous section, hypermedia data can contain redundant pieces of information, because the same piece of information can be part of more than one media object. For the following reasons we do not consider the possibility of storing information just once in a hypermedia database and thus eliminating this redundancy:

- Information contained in some medium may be contained as well in another medium. For example, information can be expressed in text or in speech, or text can even be synthesized to speech. Storing speech as text instead of audio would cause the loss of information how the text was spoken. So the information would always have to be stored in the medium with the highest information-resolution. At this moment this would be audio and video-data.

- The same information can occur in different media at different levels of detail, e.g. the video recording of the city tour contains the information when the pillar was completed while a text from a book additionally contains the information which stone-cutter finished the pillar. It is not yet possible to implement a process which translates the information into a different medium (text to audio in this case) and summarizes the information accordingly.

If one cannot eliminate the redundant representation of information, the next best option is to control and express the redundancy explicitly in the hypermedia database.

7.1.3 Corresponding Information Content

The nature and the redundancy of hypermedia data as discussed in the previous sections can be modeled by the concept of Corresponding Information Content (CIC). Corresponding Information Content relates the amount and quality of the pieces of information in one Conceptual Item to the pieces of information in another Conceptual Item.

CIC is expressed by CIC-relationships. From our point of view both conceptual items participating in a relationship occurrence represent some pieces of information. The relationship between the two conceptual items can be classified along two dimensions:

content: which information is represented, does one of the conceptual items contain more pieces of information than the other

level of detail: at which level of detail are the pieces of information represented

This classification can be used when browsing a hypermedia database not only to see if two conceptual items are related but also to see how they are related. This additional information is useful for both, the application-designer producing a new presentation, who can use the semantic information to locate and integrate relevant information, and for the user exploring existing media objects.

One data model which has a similar concept is the HDM model [Garzotto and Paolini, 1991; Garzotto *et al.*, 1993d; Garzotto *et al.*, 1993b], which offers a structured way to build multimedia applications. HDM offers a construct called "perspective". A perspective connects different representations of the same content, for example a video can be combined either with a German or an Italian audio-track, where each of these audio-tracks contains the same information. According to our understanding of the work of Garzotto et al. [Garzotto *et al.*, 1993d] the main difference to our concept is the rather coarse granularity (whole media objects versus single pieces of information) and the restriction to represent the same information at the same level of detail.

Another concept, which has been developed independently, is the "semcon" as presented in [Grosky, 1997]. A semcon is an object which is similar to an anchor. A semcon is used to represent a real world object, and is primarily used for similarity matching over multimedia data. The real world objects are connected via an appearing-in relationship to media objects. There is no direct relationship between semcons and multimedia data. The approach is severely limited because the modeling is restricted to "represents" and "appearing-in" relationships, thus no semantic modeling or temporal modeling is possible.

7.2 CIC-Relationships

As already mentioned above, a relationship can be classified along the dimensions of "content" and "detail". For each of these dimensions a relationship is tagged with a CIC-indicator. In this section we describe the tags for the CIC-indicators "content" and "detail". Then valid combinations of CIC-indicators and inverse relationships are discussed. Finally, we show that in many cases it is possible to infer tags of CIC-relationships from tags of existing CIC-relationships.

As an example for this section consider a part of a hypermedia database containing touristical information about Linz (see Figure 7.1): The logical medium on the left represents a book about Linz [Pillwein, 1824], the quote in the figure is taken from page 70. The logical medium in the middle represents a flyer issued by the local tourist board [Fremdenverkehrszentrale Linz, 1997] whereas the logical medium on the right represents the soundtrack of a video of a tour through the city.

CIC-relationships are shown as relationship occurrences between the objects of the content schema. Note: only the relationship occurrences relevant for the example are shown. Tags are represented as annotations to relationship occurrences in the form of (content-tag/level-tag). While relationships are undirected, in the example of this chapter (see Figure 7.1) relationship occurrences which are tagged with CIC-indicators are drawn as directed relationships. The tags for the inverse direction can be derived easily from the given tags as discussed in Section 7.2.4. Section 7.3 discusses how these directed relationships can be implemented in the HyperD architecture.

In Figure 7.1 only the first letter of each tag is shown. Note: pieces of information are not shown in Figure 7.1.

7.2.1 Content

The CIC-indicator "content" describes how the source and the target of a relationship occurrence are related with respect to the "amount" of information they represent.

identical: The participating conceptual items describe *exactly the same pieces* of information. In Figure 7.1 IC-objects "IC Completion Pillar 1", "IC Com-

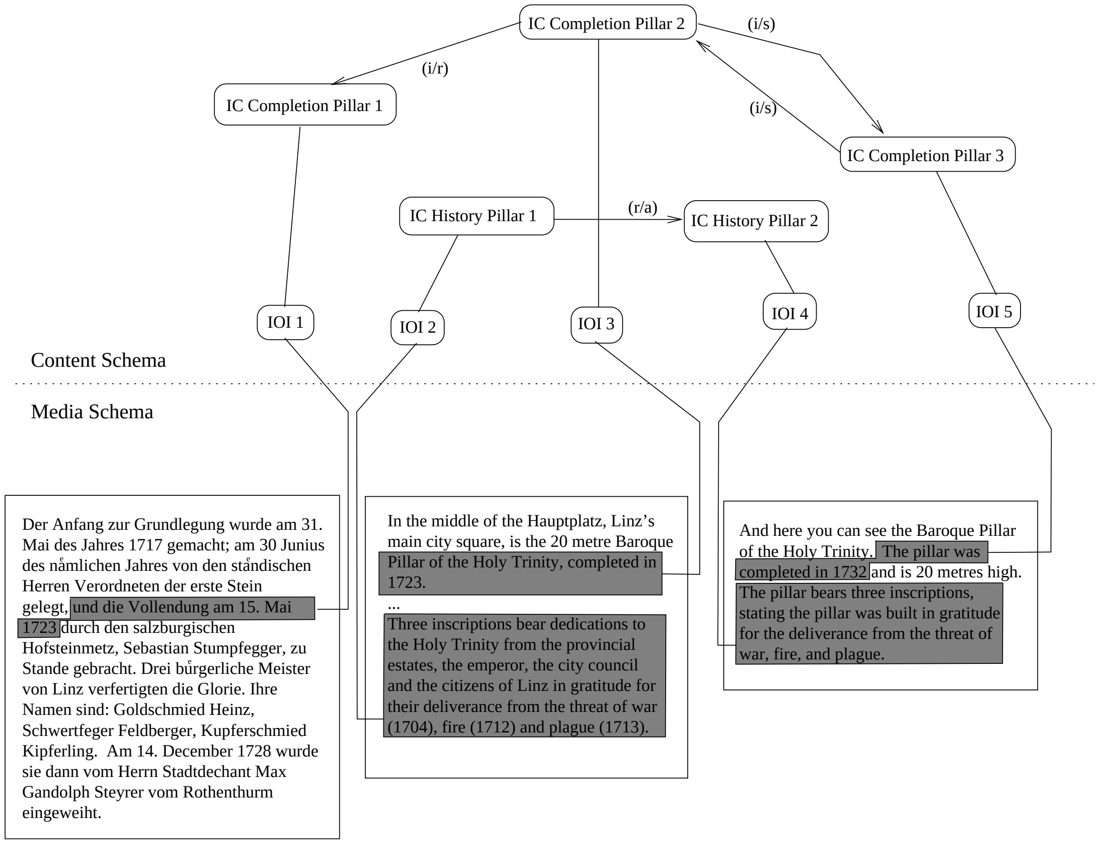

Figure 7.1: Example CIC-Relationships

pletion Pillar 2”, and “IC Completion Pillar 3” all contain the information when the pillar was completed, so the relationship occurrences “IC Completion Pillar 3” $\rightarrow$ “IC Completion Pillar 2” and “IC Completion Pillar 2” $\rightarrow$ “IC Completion Pillar 1” are tagged with “i” for identical.

extension: The target conceptual item describes *more pieces* of information than the source conceptual item. In Figure 7.1 “IC History Pillar 2” states that the pillar was erected in gratitude for the deliverance from war, fire, and plague. “IC History Pillar 1” also states the reasons why the pillar was erected, but it additionally states who the dedications are from. Thus the relationship occurrence from “IC History Pillar 2” to “IC History Pillar 1” (not shown in Figure 7.1) would be tagged “extension”.

restriction: The target conceptual item describes *less pieces* than the source conceptual item. The relationship occurrence “IC History Pillar 1” $\rightarrow$ “IC History Pillar 2” is tagged as a restriction, because “IC History Pillar 2” does not contain the information who the dedications are from.

overlap: Both conceptual items describe the *same pieces* of information while each conceptual item adds *additional yet different pieces* of information.

different: This tag can be used to model general relationships which point to information which is different from the information represented by the source conceptual item.

The advantage of the “different” tag is that relationships which connect IC-objects sharing no pieces of information can also be modeled. To classify a relationship as “different” is especially important if values for CIC-indicators may also be missing: if no tag is given for the CIC-indicator “content” of a relationship, it is not known how the source and target conceptual item of the relationship are related; if the tag of the CIC-indicator is “different”, we know that the conceptual items contain different information.

7.2.2 Level of detail

The CIC-indicator "level of detail" describes how the source and the target of the relationship are related with respect to the depth of information they represent.

same: The relationship points to an IC-object which describes the pieces of information represented by the source conceptual item at the *same level of detail*. "IC Completion Pillar 2" and "IC Completion Pillar 3" represent information about the completion of the pillar at the same level of detail.

refinement: The target of the relationship describes each piece of information which is also represented by the source *more detailed*. Whereas "IC Completion Pillar 2" only states that the pillar was completed in 1723, "IC Completion Pillar 1" gives the full date of completion. Thus the link from "IC Completion Pillar 2" to "IC Completion Pillar 1" is tagged "r" for refinement.

abstraction: The target of the relationship describes each piece of information which is also represented by the source *less detailed*. "IC History Pillar 1" represents the information about the threats to the city more detailed than "IC History Pillar 2", because in "IC History Pillar 1" also the years those threats occurred are given. The CIC-relationship occurrence from "IC History Pillar 1" to "IC History Pillar 2" is classified as "abstraction".

convex: The target of the relationship describes some pieces of information represented by the source conceptual item on a higher and some pieces of information on a lower level of detail.

different: This tag can be used to model general relationships which point to information which is different from to the information represented by the source conceptual item.

7.2.3 Combination of CIC-indicators

With the exception of the "different" tags, which can only be paired with another different tag, all other combinations of CIC-indicators are possible. If the content-tag is not "identical", the "level of detail" indicator is only valid for the part of the information which is represented by both, the source and the target conceptual item.

For example, the relationship occurrence from "IC History Pillar 1" to "IC History Pillar 2" has the level of detail tag "abstraction", which is only valid for the information about the threats to the city.

7.2.4 CIC-tags of Relationships

In the example, only CIC-tags for one direction of the relationships are shown. It is easy to deduce the CIC-tags for the inverse direction from the CIC-tags of the given direction: The content tags "identical", "overlap", and "different" stay the same, whereas an "extension"-tag is replaced with a "restriction"-tag and vice versa. The same holds true for the level of detail tags, where only "abstraction" and "refinement" tags have to be exchanged.

For example, as the relationship occurrence from "IC History Pillar 1" to "IC History Pillar 2" classified as "(restriction/abstraction)", the relationship from "IC History Pillar 2" to "IC History Pillar 1" would be classified as "(extension/refinement)".

7.2.5 Inference of CIC-tags

In some cases, it is possible to infer the tags of a relationship occurrence between two conceptual items if the target conceptual item can be reached from the source conceptual item via an alternative path: In the example there is a relationship occurrence from "IC Completion Pillar 3" to "IC Completion Pillar 2" and a relationship occurrence from "IC Completion Pillar 2" to "IC Completion Pillar 1". The tags for the relationship occurrence from "IC Completion Pillar 3" to "IC Completion Pillar 1" (not shown in Figure 7.1) can be inferred as follows:

content: "IC Completion Pillar 3" → "IC Completion Pillar 2" is tagged "identical" and "IC Completion Pillar 2" → "IC Completion Pillar 1" is also tagged "identical". It is easy to see that the derived relationship occurrence is also tagged "identical".

level of detail: "IC Completion Pillar 3" → "IC Completion Pillar 2" is tagged "same" and "IC Completion Pillar 2" → "IC Completion Pillar 1" is tagged "refinement". As "IC Completion Pillar 3" describes the pieces of information at the same level of detail as "IC Completion Pillar 2" and "IC Completion Pillar 2" describes the pieces of information less detailed than "IC Completion Pillar 1", so "IC Completion Pillar 3" also describes the pieces of information less detailed than "IC Completion Pillar 1". The derived relationship occurrence "IC Completion Pillar 3" → "IC Completion Pillar 1" is tagged "refinement".

Table 7.1 shows all possible combinations of tags for the CIC-indicator "content" and has to be read as follows: if a conceptual item A has a relationship classified as (row entry) to conceptual item B and conceptual item B has a relationship classified as (column entry) to conceptual item C then conceptual item A has a relationship to conceptual item C classified as (table entry).

When the CIC-tag of a relationship is inferred from the CIC-tags of two relationships which are followed consecutively, it is not always possible to identify a single value for the CIC-indicator, as several possibilities may exist. For example, if an extension is followed by an overlap, the resulting CIC-indicator is either extension, overlap, or different. In the cases where a "?" is shown in the tables it is not possible to infer the tag from the tags of the relationships.

To avoid indicating several possible tags, in Table 7.1 "o^+" that stands for an overlap, which could also be an extension or a refinement. When a relationship, which has been tagged as "o^+", is used to infer the tags of another relationship, the entries for "overlap" in Table 7.1 have to be used to look up the result. Similarly, the table for the level-tags includes an additional tag "c^+".

Table 7.2 shows the inferred "level of detail"-tags of two relationships followed consecutively, if the derived content-tag is not "?".

	identical	extension	restriction	overlap	different
identical	identical	extension	restriction	overlap	different
extension	extension	extension	?	?	different
restriction	restriction	o^+	restriction	o^+	?
overlap	overlap	o^+	?	?	?
different	different	?	different	?	?

Table 7.1: Combinations of Tags for CIC-indicators "content"

	same	refinement	abstraction	convex	different
same	same	refinement	abstraction	convex	different
refinement	refinement	refinement	c^+	c^+	different
abstraction	abstraction	c^+	abstraction	c^+	different
convex	convex	c^+	c^+	c^+	different
different	different	different	different	different	?

Table 7.2: Combinations of Tags for CIC-indicators "level of detail"

7.3 CIC relationships in the HyperD Architecture

There are several possibilities how the CIC information, expressed through the CIC-tags, can be incorporated into the HyperD architecture:

- All conceptual relationships can carry CIC information. This solution is easy to implement and many CIC tags, for example tags of relationships between IOIs (see below), can be derived automatically. Because of the large number of CIC tags, many of the tags would remain empty or would carry automatically assigned, trivial information.

- Only relationships between ICs (Information Content Rel.) can carry CIC-tags.

  This possibility is also easy to implement and again some tags can be derived either from whole-part relationships between IOIs the ICs are connected to or from the relationships between the ICs.

We have chosen to use the first approach to implement CIC in the HyperD architecture to allow for a uniform modeling of Corresponding Information Content.

Given that all conceptual relationships can carry CIC information, the question of how to attach the CIC-tags remains. One possibility would be to derive a subclass from Conceptual Relationship where the CIC-tags are added. This solution has severe disadvantages, especially if a relationship is created without the CIC-tags. If at any later time the CIC-tags have to be added, the relationship has to change its superclass: if a subclass is derived from Conceptual Relationship without the CIC-tags and the CIC-tags have to be added, the class has to be subclasses from a different class, possibly along with all classes of the inheritance hierarchy between the changing class and the Conceptual Relationship class.

For this reason we have chosen a different approach: the CIC-tags are added directly to class Conceptual Relationship. This ensures a uniform treatment of all relationships and allows an easy implementation. Figure 7.2 shows the top level object model of the content schema with the added support for CIC-Relationships. Note: only additional attributes and methods are shown.

One further problem has to be addressed: Conceptual Relationship relationships are undirected relationships while CIC-relationships are directed relationships. The solution to this problem is simple: The CIC tags are retrieved with the getContentTag and getLevelTag method. Both methods take as their only parameter the OID of the Conceptual Item which is point of view (i.e. the "source" of the relationship). Internally, only one pair of tags has to be stored (contentTag and levelTag) along with the conceptual item (sourceCI) which is the "point of view" for the tags. The tags for the other direction can easily be calculated according to the rules given in Section 7.2.4.

IOI-IOI relationships are always tagged "(extension/same)" in the direction from the part-IOI to the whole-IOI. The relationship from the whole-IOI to the part-IOI is always tagged "(restriction/same)". While this does not add much information for the user of the database, the fixed tags are useful for calculating derived CIC-tags.

CIC-relationships can be added easily to the presentation schema of the HyperD architecture: only the getContentTag and getLevelTag methods have to be added to the Presentation Relationship class. The tags are derived from the tags in the conceptual schema as described in Section 7.2.5.

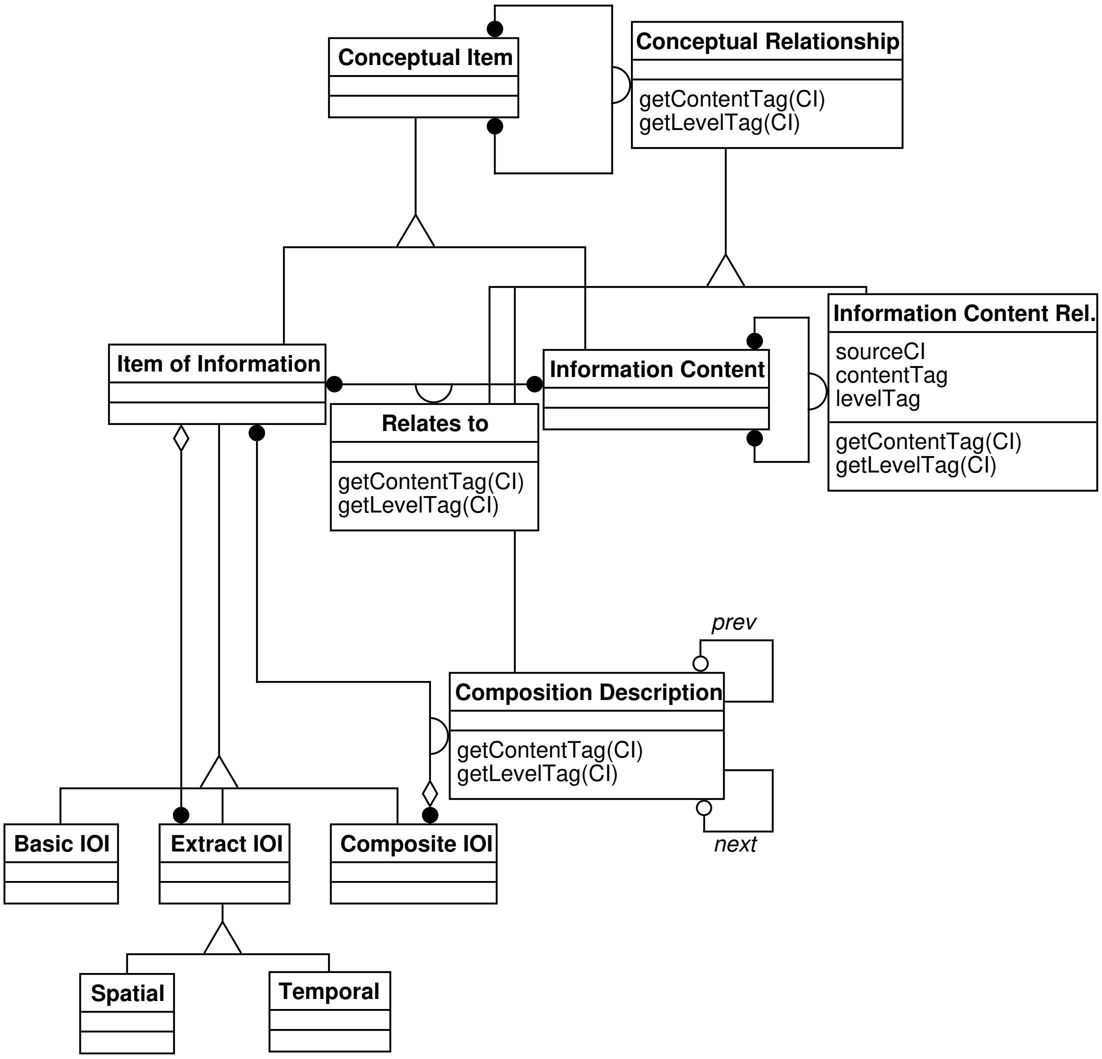

Figure 7.2: Top Level Object Model of the Content Schema With CIC-tags

Chapter 8

Prototype

This chapter describes implementation issues of a prototype implementation of selected parts of the storage and media level of the **Hyper**media **D**atabase Management System Architecture. It is shown how the media level and the content level of the HyperD architecture can be implemented on top of an existing database management system. In this chapter only the implementation for the data type image is discussed, the implementation for other multimedia data types can be done in a similar manner.

Section 8.1 describes relevant features of the Illustra database management system, a commercial version of Postgres [Stonebraker and Kemnitz, 1991], which is the basis for the implementation. Section 8.2 discusses issues of the implementation.

8.1 Illustra

This section describes the Illustra database management system and the Illustra Image DataBlade, which extends the Illustra database management system with the capabilities necessary to store and manipulate images.

8.1.1 Illustra Database Management System Server

Illustra [Illustra Information Technologies, Inc., 1994a; Illustra Information Technologies, Inc., 1994c] is an object-relational database management system. In addition to the data in the relations, functions operating on these relations can be stored in the database. The functions can be written either in SQL or in C. Illustra also offers a complete interface for client programs written in C.

Abstract data types can be implemented as types (which are mapped to relations) and functions operating on these types. Abstract data types can have custom aggregate functions, operators, constructors, and destructors. Both simple and multiple inheritance of types is supported, as well as late binding of overloaded functions and polymorphism. Illustra features special support for large objects (objects larger than 8KB), which can be distributed over several drives in a round-robin fashion.

Illustra is an active database management system and offers rules and alerters. Relations can contain virtual columns, whose values are derived or calculated. Temporal queries (what would this query have returned at May, 12^{th}) are also supported.

The capabilities of the Illustra DBMS server can be extended by adding DataBlades to the database management system. Datablades are modules containing additional data types and functions operating on these data types. Available DataBlades provide data types and functionality for the storage and the manipulation of images, text, 2D and 3D data, and enhanced temporal capabilities.

8.1.2 Illustra Image DataBlade

This section describes the Illustra Image Datablade [Illustra Information Technologies, Inc., 1994b]. The Illustra Image DataBlade provides a data type which allows to store images as attributes in relations. The DataBlade also provides commands to load images into the database (with conversion if necessary), to retrieve images, to display images (again with conversion), and also operators to manipulate the images stored in the database. The Illustra Image DataBlade is built using the pbmplus package, which is a public domain package for the conversion and the manipulation of images.

Section 8.1.2.1 describes the image formats used for the storage of images in the database management system. The operators provided by the Image DataBlade for the manipulation of images are described in Section 8.1.2.3, grouped according to the classification described in Section 8.1.2.2. Section 8.1.2.4 contains the results of performance measurements of the transformation operators provided by Illustra.

8.1.2.1 Image Formats

The Image DataBlade uses three image formats for the internal storage of images:

pbm: portable bitmap, 1 bit per pixel. This encoding is used for black and white raster images.

pgm: portable greymap, 8 bit per pixel. This encoding is used for images with up to 255 shades of grey.

ppm: portable pixmap, 24 bit per pixel. This encoding is used for truecolor images.

As Illustra can be extended easily, the DataBlade is not limited to these encodings, additional storage formats can be added.

8.1.2.2 Classification of Operators

The operators which can be used to manipulate the media data of logical media can be classified, according to the effect they have on the database, as follows:

Transformer: Yields a logical medium of the same media type. Transformers are used to instantiate secondary internal media.

Renderer: Renders a logical medium into a logical medium of a different media type. A text can be rendered into an image before combining it with another image to create an annotated image, a sequence of images can be rendered into a video. Rendering causes the creation of a new logical medium.

Combiner: Combines two logical media into one. Combinations of media of the same media type are always possible, combinations of media of different media types are only allowed if there is a class which can accommodate the resulting type. Combine-operators also create a new logical medium.

Insert: Inserts a logical medium into the database. A new logical medium, a primary internal medium, and the chunks containing the data are created.

Delete: Deletes the logical medium from the database, together with all associated internal media and media chunks from the database.

Query-function: Retrieves information about the media object.

8.1.2.3 Operators in the Image DataBlade

The Image DataBlade provides operators (called functions in the terminology of Illustra) for manipulating images. Tables 8.1 to 8.6 contain the names of the available operators as well as brief descriptions of their functionality. The descriptions are taken from the documentation (manual pages) of the pbm-plus package (pbmplus10dec91.tar.Z).

The operators are grouped according to the classification of operators described in Section 8.1.2.2. Note: The "delete" and "renderer" categories described in Section 8.1.2.2 are missing from the following list, because the Image DataBlade contains no operators belonging to those categories. The categories describe which effect an operator has on the database. In the context of the Image DataBlade, they describe how the operators affect the image or the images provided as their input:

Transformer: transforms one image into another image. Tables 8.1 and 8.2 contain the functions classified as transformers.

Combiner: combines two images into one. Table 8.3 contains the functions classified as combiner functions.

Insert: a new image is inserted into the database. Table 8.4 contains the functions classified as insert and output functions. Table 8.5 contains the functions which also insert media data into the database, but generate the data themselves.

Query-function: retrieves information about the media object. Table 8.6 contains the functions classified as query functions.

8.1.2.4 Performance Measurements

This section contains the results of an experiment to measure the performance of the transformation operators provided by the Illustra Image DataBlade. The goal is not to give the exact duration of each operation, but to have an overall assessment on how much time is saved by avoiding the execution of an operator, i.e. if it is beneficial to have an additional secondary medium, and to have a basis for a cost-function for query optimization (see Section 8.2.6).

As the timing of operators inside the database is difficult, the measurements were carried out outside the database management system, using the programs

Operator	Description
all images	
convert	convert an image into the desired format
convolve	general MxN convolution on an image
crop	crop all like-colored borders off an image
depth	change the maxval in an image
enlarge	enlarge an image
fliplr	flip an image left to right
fliptb	flip an image top to bottom
gamma	perform gamma correction on an image
invert	invert an image
margin	add a margin to an image
rotate	rotate an image
scale	scale an image
shear	shear an image
tile	replicate an image into a specified size
transpose	flips an image top-to-bottom and left-to-right
trunc	select a rectangular region from an image

Table 8.1: Transformers: All Images

Operator	Description
pixmap images	
dither	ordered dither for color images
quantize	quantize colors down to a specified number
relief	run a Laplacian Relief filter on an image
greymap images	
bentley	Bentleyize an image
edge	edge-detect an image
enhance	edge-enhance an image
oil	apply a transformation so that the image looks like an oil painting
normalize	normalize the contrast of an image
bitmap images	
life	apply Conway's rules of life to an image
mask	create a mask bitmap from an image

Table 8.2: Transformers: Pixmap, Greymap, and Bitmap

Operator	Description
paste	paste an image into another image
And_op	combine two images using the logical and operation
Or_op	combine two images using the logical or operation
Xor_op	combine two images using the logical xor operation
Plus	perform addition on two images
Minus	perform subtraction on two images
Times	perform multiplication on two images
concat	concatenate images

Table 8.3: Combiners

Operator	Description
insert	copy an image into the database
GetImg	extract an image into a file
ImgDisplay	display an image on the screen
toascii	read a raw image and convert it to a non-raw image

Table 8.4: Input/Output

Operator	Description
upc	create a Universal Product Code bitmap
crater	create cratered terrain by fractal forgery
ramp	generate a greyscale ramp
pattern	create a pretty pixmap

Table 8.5: Generators

Operator	Description
GetColor	get the color value of a point of an image
GetHeight	get the height of an image
GetImgType	get the internal format of an image
GetImgMaxColor	get the maximum color value in an image
GetImgSize	get the size of an image
GetOrigPath	get the path of the image input file
GetOrigFormat	get the format of the image input file
GetOrigType	get the original format of an image
GetWidth	get the width of an image
histogram	calculate a color histogram and write it to a file

Table 8.6: Query Functions

of the functionally equivalent pbm-plus package (pbmplus10dec91.tar.Z). There might be slight differences to the internal operators, but these differences can be neglected for the purposes stated above. The measurements were performed on a Sparc 4 workstation (running Solaris 2.5).

For the benchmark one truecolor image (ppm), one greycolor image (pgm), and one bitmap image (pbm) were used. Each image was used in three sizes, each size with half the number of pixel than the next bigger image. For the greycolor image, the truecolor image was demoted to a greymap image. Table 8.7 describes the properties of the images, which are shown in Figure 8.1.

The benchmark was conducted as follows: each function which has been classified as a transformer function in Section 8.1.2.3 is applied to each of the nine images described above. Those functions which require the image to be of a certain type were only tested with compatible images. First each function/image combination was performed 5 times—although one time would have been sufficient—to ensure that all images and programs were loaded into the cache. Then the same function/image combination was executed 10 times to perform the benchmark. One complete run of all function/image combinations lasted about 6 hours. Ten of those runs have been performed, giving one hundred measurements for each function/image combination. All runs were performed over the weekend or during the night to ensure a low system load without user processes.

The results of the benchmark for the bitmap operators are shown in Table 8.8. Tables 8.9 and 8.10 contain the results for the greymap operators, Tables 8.11 and 8.12 contain the results for the pixmap operators, and Tables 8.13 to 8.20 contain the results for the anymap operators which can be applied to images of any kind. Note: The entry for ppm3.ppm for command ppmhist in Table 8.11 is not an error, the result seems to be caused by a bug in ppmhist.

The time given in the tables is the sum of the "User CPU time" and the "System CPU time" as reported by the `/usr/bin/time` command.

Filename	Fileformat	Filesize (bytes)	Colors	Size (pixel)	Pixel
ppm1.ppm	ppm	3240016	15886	1200x900	1080000
ppm2.ppm	ppm	1617999	12663	848x636	539328
ppm3.ppm	ppm	810015	10889	600x450	270000
pgm1.pgm	pgm	1080016	256	1200x900	1080000
pgm2.pgm	pgm	539343	256	848x636	539328
pgm3.pgm	pgm	270015	256	600x450	270000
pbm1.pbm	pbm	135012	2	1200x900	1080000
pbm2.pbm	pbm	68277	2	848x636	539328
pbm3.pbm	pbm	33761	2	600x450	270000

Table 8.7: Images used for the Benchmark

The "User CPU time" is the sum of the tms_utime and tms_cutime fields returned by the times(2) function. The tms_utime is the CPU time used while executing instructions in the user space of the calling process, the tms_cutime is the sum of the tms_utimes of all child processes (see also the manual page of times(2)).

The "System CPU time" is the sum of the tms_stime and tms_cstime fields returned by the times(2) function. The tms_stime is the CPU time used by the system on behalf of the calling process. The tms_cstime is the sum of the tms_stimes of all child processes (see also the manual page of times(2)).

The time is given in seconds, the resolution of the `/usr/bin/time` command is 1/100 second.

As shown in this section, the execution of operators consumes considerable time, thus avoiding the execution by providing secondary internal media is beneficial. The results presented in this section are also used to construct the cost function described in Section 8.2.6.

pbm1.pbm pbm2.pbm pbm3.pbm

pgm1.pgm pgm2.pgm pgm3.pgm

ppm1.ppm ppm2.ppm ppm3.ppm

Figure 8.1: Images used for the Benchmark

Size	pbmlife	pbmmask
1200x900	2.54	2.92
848x636	1.46	1.66
600x450	0.90	1.02

Table 8.8: Bitmap Operators

Size	pgmbently	pgmedge	pgmenhance	pgmhist
1200x900	2.09	10.99	3.97	1.11
848x636	1.18	5.63	2.15	0.78
600x450	0.76	2.97	1.24	0.62

Table 8.9: Greymap Operators 1

Size	pgmnorm	pgmoil	pgmramp
1200x900	1.91	73.33	4.20
848x636	1.13	36.81	1.32
600x450	0.73	18.56	2.28

Table 8.10: Greymap Operators 2

	ppmdither			ppmhist			ppmpat
Size	pbm	pgm	ppm	pbm	pgm	ppm	ppm
1200x900	11.07	11.40	11.98	5.01	5.34	7.56	8.10
848x636	5.71	5.86	6.15	2.70	2.86	6.00	4.11
600x450	3.02	3.12	3.27	1.58	1.68	180.02	2.18

Table 8.11: Pixmap Operators 1

Size	ppmquant			ppmrelief		
	pbm	pgm	ppm	pbm	pgm	ppm
1200x900	10.72	11.09	27.46	3.65	4.04	4.54
848x636	5.56	5.73	15.44	1.98	2.11	2.42
600x450	2.95	3.07	9.36	1.13	1.20	1.37

Table 8.12: Pixmap Operators 2

Size	pnmarith								
	-add			-multiply			-subtract		
	pbm	pgm	ppm	pbm	pgm	ppm	pbm	pgm	ppm
1200x900	2.95	3.72	5.93	4.19	5.01	10.01	3.06	3.75	5.97
848x636	1.63	2.03	3.07	2.27	2.68	5.09	1.71	2.05	3.09
600x450	1.00	1.19	1.66	1.32	1.52	2.69	1.04	1.24	1.68

Table 8.13: Anymap Operators 1

Size	pnmcat			pnmconvol			pnmcrop		
	pbm	pgm	ppm	pbm	pgm	ppm	pbm	pgm	ppm
1200x900	3.94	5.12	8.09	23.05	22.74	37.25	1.91	2.51	3.97
848x636	2.10	2.72	4.22	11.74	11.48	18.58	1.13	1.43	2.16
600x450	1.18	1.52	2.28	5.89	5.89	9.34	0.74	0.89	1.26

Table 8.14: Anymap Operators 2

Size	pnmcut			pnmdepth			pnmenlarge		
	pbm	pgm	ppm	pbm	pgm	ppm	pbm	pgm	ppm
1200x900	0.52	0.59	0.73	2.32	2.55	3.98	5.97	7.83	12.03
848x636	0.48	0.52	0.62	1.33	1.45	2.14	2.77	3.83	6.00
600x450	0.44	0.48	0.55	0.84	0.91	1.27	1.50	2.03	3.06

Table 8.15: Anymap Operators 3

Size	pnmflip								
	-leftright			-topbottom			-transpose		
	pbm	pgm	ppm	pbm	pgm	ppm	pbm	pgm	ppm
1200x900	3.41	4.22	5.81	5.90	6.51	7.89	6.18	6.79	8.21
848x636	1.89	2.23	3.01	3.16	3.45	4.11	3.23	3.60	4.24
600x450	1.11	1.30	1.66	1.74	1.90	2.23	1.78	1.93	2.26

Table 8.16: Anymap Operators 4

Size	pnmgamma			pnminvert			pnmmargin		
	pbm	pgm	ppm	pbm	pgm	ppm	pbm	pgm	ppm
1200x900	2.28	2.58	3.98	2.35	2.76	4.25	5.90	7.60	11.42
848x636	1.34	1.47	2.15	1.34	1.56	2.27	4.10	5.00	6.91
600x450	0.85	0.92	1.25	0.87	0.95	1.30	3.24	3.68	4.66

Table 8.17: Anymap Operators 5

Size	pnmpaste					
	-and	-or	-replace			-xor
	pbm	pbm	pbm	pgm	ppm	pbm
1200x900	1.45	1.45	1.46	2.12	3.53	1.45
848x636	0.92	0.92	0.94	1.26	1.92	0.91
600x450	0.64	0.64	0.65	0.81	1.15	0.65

Table 8.18: Anymap Operators 6

Size	pnmrotate			pnmscale			pnmshear		
	pbm	pgm	ppm	pbm	pgm	ppm	pbm	pgm	ppm
1200x900	22.49	22.22	38.94	5.90	6.07	16.27	6.24	6.57	12.30
848x636	11.19	11.22	19.41	3.09	3.16	8.24	3.26	3.34	6.30
600x450	5.62	5.66	9.77	1.70	1.75	4.31	1.81	1.84	3.28

Table 8.19: Anymap Operators 7

Size (resulting image)	pnmtile		
	pbm	pgm	ppm
3600x2700	33.42	36.73	45.57
2544x1908	16.25	18.25	22.21
1800x1350	8.00	9.06	11.30

Table 8.20: Anymap Operators 8

8.2 Implementation

This section describes the implementation of the storage level and of the media level of the HyperD architecture using the Illustra Image Datablade.

Section 8.2.1 describes the mapping from the design of the HyperD architecture as described in Chapter 5 to the implementation using Illustra. Sections 8.2.2 and 8.2.3 describe how the objects of those two levels are created and deleted. The final three sections deal with query execution and query optimization.

8.2.1 Implementation Decisions

Illustra is an extended relational database management system, thus the data is stored in relations, where each relation can have a data type. Each class of the storage and of the media schema is mapped to a type; its extension, i.e. the set of its instances, is represented by a relation of that data type. Each instance of a class is stored as one tuple of the corresponding relation. Methods are mapped to functions, whose first parameter is the OID (Object IDentifier) of the object on which the method is executed. Illustra supports inheritance and late binding of functions, which is used to implement the inheritance hierarchies shown in the object models of Chapter 5.

Each piece of multimedia data is stored as an attribute in a relation which implements a subclass of class Chunk. For example, each tuple of the relation containing the ImageChunk objects has an image attribute. The domain of this attribute is derived from the image-type provided by the Illustra Image DataBlade. Externally provided images are converted into an internal format (portable bitmap/greymap/pixmap) and then stored as an attribute of type image.

8.2.2 Insertion of New Media Objects

Insertion is handled via conventional insert, i.e. by using an overloaded function for the SQL insert command. The insertion of a new logical medium causes:

- The creation of the logical medium.
- The creation of the primary internal medium.
- The creation of a media chunk containing the original image data.

The distinction between internal media and media chunks does not seem to be necessary for storing images, because, at least in our examples, there is only one chunk for each internal image. Internal media have been introduced to conveniently model media types whose objects can consist of several chunks, e.g. audio and video objects.

8.2.3 Creation and Deletion of Secondary Internal Media

After the creation of the logical medium, secondary internal media can be instantiated for those queries, which the database administrator expects to be requested frequently.

The function `addSecondary(internalMedium, operatorList)` is used to create a secondary internal medium. The first parameter is the OID of the internal medium which serves as the starting point for the creation. The operators contained in the second parameter are applied to the media chunks of the internal medium selected as the starting point. The resulting media chunks are connected to the new secondary internal medium.

The OID of an internal medium is retrieved using the function `getInternalMedium(logicalMedium, operatorList)`. The first parameter is the OID of the logical medium. The second parameter is the sequence of operators used to create the internal medium. An empty list selects the primary internal medium itself. If the requested internal medium does not exist, a null value is returned.

We allow more than one operator between internal media, because otherwise an internal medium would have to be created after each application of an operator. While this is still possible, we allow lists of operators to be able to avoid an excessive number of unused secondary internal media and the accompanying wasting of disk space.

The internal media are organized in a directed acyclic graph, where alternative paths to a secondary internal medium cannot be instantiated (i.e. only one path can lead from the primary internal medium to a given secondary internal medium): if the user wants to create the paths `op1,op2` and `op2,op1`, two secondary internal media have to be created. Usually this is not necessary, because during query execution the path `op2,op1` can be transformed into `op1,op2` (and vice versa, see section 8.2.6) if the resulting image for both paths is the same.

Secondary internal media can be removed using the function `removeSecondary(internalMedium)`.

The following function calls generate the internal media shown in Figure 5.3:

1. `addSecondary(getInternalMedium(LM, ""), "quant(255)")`
   The request causes the instantiation of the secondary internal medium S0. The function `getInternalMedium` returns the OID of the primary internal medium of the logical medium `LM`. Then the operator "quant(255)" reduces the number of colors in the image of the primary internal medium, which can contain up to 16.777.216 colors (3 bytes per pixel) to 255 colors, which can be displayed by a large number of display devices. The resulting image is stored as a new image chunk connected to S0.

2. `addSecondary(getInternalMedium(LM, "quant(255)"),`
   `"scale(640,480)")`
   The function `getInternalMedium` returns the OID of S0. Then the scale operation is applied to the image of S0 resulting in S1, thus S1 represents an image of 640x480 pixel and 255 colors.

3. `addSecondary(getInternalMedium(LM, "quant(255)"),`
   `"trunc(400,510,540,670), margin(20)")`
   This expression creates the secondary internal medium S2 by applying the trunc operator followed by the margin operator to the image of S0.

Secondary internal media are used to accelerate query execution as shown in the following sections.

8.2.4 Retrieval of Media Objects

Concrete representations of a logical medium are retrieved using the function `retrieve(logicalMedium, operatorList)`.

The function returns the OID of an internal medium:

- If the query matches an internal medium, the OID of the internal medium is returned.
- If the query does not match an internal medium, a temporary internal medium is created and its OID is returned.

The user can delete the returned temporary internal medium using the function `removeTemporarySecondary(internalMedium)`. If the function is called with the OID of a regular internal medium, i.e. an internal medium which is not a temporary internal medium, the function does nothing.

8.2.5 Commutativity of Operators

Some operators are commutative with other operators. This property is important during query execution, described in the next section, where two ordered lists of operators are compared.

Table 8.21 shows the commutativity matrix for the relevant operators of the Illustra Image DataBlade. Operators are commutative, if their position in the query can be swapped without changing the result of the query, i.e. the resulting image is the same in both cases. The table has to be read as follows: op1 and op2 can be swapped, if the cell in row(op1) and column(op2) contains a "+" sign. If the cell contains a"−"sign swapping is not allowed, because the resulting image would be different. The cells containing a "±" sign represent a special case: swapping these operators can cause slight differences in the resulting image. For example, for most images it is not important if first the trunc and then the normalization operation is performed. But if the extracted part contains significantly less colors than the whole image, then there will be a difference in the resulting image. It is up to the database administrator to balance the quality of the resulting images against speed and storage efficiency by either storing a

"+" or a "−" sign in the cell. Note: swapping operators may cause changes in the parameters. For example, if trunc is swapped with enlarge, the parameters of trunc have to be adjusted.

Inheritance is used to define commutativity for new data types. The top left quadrant of Table 8.21 shows the operators which can be applied to all images. This part of the table can be shared by all image types (bitmap, greymap, and pixmap images). The table can be extended for different subtypes of images. Table 8.21 also shows the extension for pixmap images. Only the entries for the operators dither, quantize, and relief, which are specific to pixmap images, have to be added.

The commutativity table can be implemented as a relation with two attributes containing each pair of operators which has a plus in Table 8.21. When optimizing, two operators can be swapped if (op1, op2) is a tuple of the relation. Thus new operators can be added simply by inserting appropriate pairs of operators that are commutative.

8.2.6 Execution of Queries

Queries, sequences of operators, are used to obtain concrete representations of a logical medium (see Section 8.2.4). As a default, such representations are created on the fly. To accelerate the execution of a query, secondary internal media are used. These secondary internal media are either concrete variations which are expected to be used frequently or variations of the media object to which certain (time-expensive) operators have already been applied.

The query execution consists of the following steps:

1. Find the set containing all internal media which can be transformed into the result of the query. For each internal medium belonging to the logical medium the commutativity table is used to check whether the operators

	general operators																pixmap operators		
	convert	convolve	crop	depth	enlarge	fliplr	fliptb	gamma	invert	margin	rotate	scale	shear	tile	transpose	trunc	dither	quantize	relief
convert	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
convolve	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
crop	-	-	+	+	+	+	+	±	+	-	+	+	+	-	+	+	-	±	-
depth	-	-	+	-	±	+	+	-	-	±	+	±	±	+	+	+	-	±	-
enlarge	-	-	+	±	+	+	+	+	+	+	+	+	+	-	+	+	-	+	±
fliplr	-	-	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
fliptb	-	-	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
gamma	-	-	±	-	+	+	+	-	-	±	+	±	±	+	+	+	-	-	-
invert	-	-	+	-	+	+	+	-	+	+	+	+	+	+	+	+	-	-	-
margin	-	-	-	±	+	+	+	±	+	+	+	+	-	-	+	+	-	±	±
rotate	-	-	+	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+
scale	-	-	+	±	+	+	+	±	+	+	+	+	+	-	+	+	±	±	+
shear	-	-	+	±	+	+	+	±	+	+	-	+	+	-	+	-	±	±	±
tile	-	-	-	+	-	-	-	+	+	-	-	-	-	+	-	-	+	+	+
transpose	-	-	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+
trunc	-	-	-	+	+	+	+	+	+	+	+	+	-	-	+	+	+	+	+
dither	-	-	-	-	-	+	+	-	-	-	+	±	±	+	+	+	+	-	-
quantize	-	-	±	±	+	+	+	-	-	±	+	±	±	+	+	+	-	-	-
relief	-	-	-	-	±	+	+	-	-	±	+	+	±	+	+	+	-	-	+

Table 8.21: Commutativity of Operators for Images and Pixmap Images

contained in the operatorList (L_1) of the query can be transformed into the operators which were used to generate the internal medium (L_2). This check proceeds in the following steps numbered (a) to (e):

(a) If $set(L_2) \not\subseteq set(L_1)$—where $set(L)$ is the set of operator names used in the operatorList L—terminate, the lists are not compatible.

(b) $i = 1$

(c) Locate the i^{th} operator of L_2 in L_1 and try to bring the operator to the i^{th} position in L_1 by swapping the operator with the preceding operators if the corresponding cell in Table 8.21 contains a "+" sign. If this fails or if the operators at the i^{th} position have different parameter values after the swapping, terminate, the lists are not compatible.

(d) Repeat step (c) with with $i = i + 1$ until the end of L_2 is reached.

(e) Terminate, the lists are compatible. If $L_1 = L_2$ the internal medium already contains the result of the query, which can be returned immediately, otherwise the internal medium is added to the set of possible starting points.

2. Select the best internal medium: For each of the possible internal media the cost to transform the internal medium into the result as specified by the query is calculated and the internal medium with the lowest cost is selected.

   The cost is calculated using the function `calculateCost(internalMedium, operatorList)`. The second parameter is the `operatorList` passed to `retrieve`. `calculateCost` returns the estimated time for the execution of all operators of the `operatorList` which have not yet been applied to the internal medium.

   A simple cost function can be based on the measurements of Section 8.1.2.4: for many operators, the execution time depends on the type of image (bitmap, greymap, or anymap) and is proportional to the size of the image. So a very simple cost function could be:

$$c = s \times c_1 \times c_2$$

where c is the cost, s is the size of the image in pixel, c_1 is the cost of executing the operation for one pixel, and c_2 is the additional cost for executing the operation in a higher image format (if applicable). c_2 is only necessary if the operation can deal with more than one image format, otherwise it is 1.

3. Perform the query: The selected internal medium, i.e. the internal medium with the lowest cost, is, if necessary, transformed into a temporary secondary medium and then the result is returned.

Examples:

`retrieve(LM, "scale(640, 480), quant(255)")`:

First the set of possible starting points is determined: the primary internal medium is always an element of the set.

According to Table 8.21 the query expression can be transformed into "quant(255), scale(640, 480)". The first operator is the same as the operator used to create S0, thus S0 is added to the set. S1 is also added, because the operators of S1 and the query operators are the same.

Then the "cheapest" internal medium is selected, which is obviously S1 because no additional processing is necessary.

The performance gain for this query is 51 seconds on our Sparc 4.

`retrieve(LM, "quant(255), trunc(400, 510, 540, 670)")`:

In this case the set of possible starting points consists of the primary internal medium and S0. S0 is chosen as the starting point, because only the operation `trunc(400, 510, 540, 670)` has to be executed, which is faster than executing both operations on the primary internal medium.

The performance gain for this query is 41 seconds on our Sparc 4.

As demonstrated with the last query, there is no need to instantiate secondary internal media for all requested queries. Not only the application for which the secondary internal medium was created benefits from the secondary internal medium, but every application which can use the secondary internal medium as a starting point to create the desired variation. The performance gain is achieved automatically without the need to change any application or the database schema. Deletions of secondary internal media also do not result in changes to applications or to the schema, although the execution time of certain queries increases.

Storage space can be saved if many different queries "share" one secondary internal medium representing an intermediate stage of the queries instead of having one secondary internal medium for each query.

In contrast to conventional database management systems, instantiating additional secondary internal media does not cause additional performance penalties due to more expensive update operations, because we do not allow updates to logical media which change the physical data of the primary internal medium. The disadvantages of having too many secondary internal media are an increased demand of storage space and an increase of the time needed to determine the internal medium which is used to compute the result of a query.

Chapter 9

Conclusion

In this thesis we have presented the **Hyper**media **D**atabase Management System (HyperD) Architecture for hypermedia database management systems supporting data independence and the classes needed to implement the architecture.

We have shown that the concept of physical and logical data independence can be applied beneficially to hypermedia database management systems. We believe that the HyperD architecture can be extended easily for other media types, e.g. graphics (built from primitives like lines and areas in contrast to images) or midi-streams, and that the HyperD architecture is flexible enough to model domain specific information.

The approach to physical data independence presented here takes the characteristics of multimedia data into account and allows a controlled reuse of multimedia objects in an unlimited number of applications without the need of redundant storage, because the requested variation of the multimedia object can be calculated on the fly. Benchmarks show that the additional complexity imposed on the design by the inclusion of secondary internal media is justified, because the improvements of the performance can be dramatic, depending on which operators are used in the queries. Secondary internal media are used transparently and automatically, i.e. the applications do not need to know about changes regarding secondary internal media. Thus true physical data independence is achieved.

The approach to logical data independence presented in this thesis is highly flexible and allows to represent links and nodes without any additional class structure. But it is also possible to use classes to structure the links and nodes and to include classes in the presentation.

The HyperD architecture allows to model the structure as well as the content of hypermedia data in a consistent, highly flexible, object oriented manner. In particular, we have shown that whole-part relationships based on composition as well as whole-part relationships based on decomposition are needed to capture the full meaning of multimedia data in the context of hypermedia database management systems. We have shown that temporal as well as spatial relationships can be modeled in an orthogonal manner. We have discussed the characteristics of these relationships and how they can be embedded into the content level of a hypermedia database management system.

Important advantages include an integrated approach to temporal and spatial modeling, the explicit incorporation of decomposition as a construct, the orthogonal treatment of extracts and composites, a flexible and powerful way to incorporate multimedia data into a semantic hypermedia network by providing flexible propagation of relationships along whole-part hierarchies, and an easier data management due to the automatic handling of inserts and updates of dependent parts in whole-part relationships.

This thesis introduces the concept of Corresponding Information Content (CIC) and shows how the concept can be implemented in the HyperD architecture. CIC allows to model the amount and quality of the pieces of information in one media object in relationship to the pieces of information in another media object.

We have described a prototypical implementation for images, other media types can be implemented in the same manner. The implementation of a proof of concept prototype using Illustra, based on the concepts presented in this thesis, is the topic of a current master thesis [Lisberger, 1998], supervised by the author of this thesis. The preliminary results of the master thesis—the implemementation has been completed—show that the architecture presented here is sufficient to build a highly flexible hypermedia database management system.

Further and open work includes the following issues:

- Full specification of all methods and attributes of the classes of the HyperD architecture.
- Definition of a full query language along with the definition of query transformation strategies.
- Full integration of media objects containing more than one kind of media data (see Section 5.3.3).
- Integration of existing multimedia indexing techniques and information retrieval techniques as discussed in Section 3.1.2.
- The implementation of the HyperD architecture for additional datatypes.

List of Figures

1.1 Highlevel Hypermedia DBMS Architecture 9

2.1 OMT: Classes 33
2.2 OMT: Objects 34
2.3 OMT: Relationships 35
2.4 OMT: Multiplicity 36
2.5 OMT: Example 37
2.6 OMT: Aggregation 38
2.7 OMT: Inheritance 38
2.8 OMT: Relationship Inheritance 39

4.1 Extract of the ANSI/X3/SPARC Architecture 99

5.1 High Level Object Model 110
5.2 Object Model of the Storage Schema 114
5.3 Example Internal Media 117
5.4 Object Model of the Media Schema 119
5.5 Object Model of the Content Schema 124
5.6 Example Content Schema 126
5.7 Detail from the Object Model of the Content Schema: Interface to the Media Schema 127
5.8 Detail from the Object Model of the Content Schema: Composition 133
5.9 Example of a Temporal Inconsistency 136
5.10 Placement of Extracts 139
5.11 "Yo-yo Effect" 143
5.12 Object Model of the Presentation Schema 150
5.13 Example Presentation Schema 152

5.14 Detail from the Object Model of the Content Schema: Presentation Items 154
5.15 Object Model of the Meta Schema 159

6.1 DBMS Architectures 164

7.1 Example CIC-Relationships 177
7.2 Top Level Object Model of the Content Schema With CIC-tags . 185

8.1 Images used for the Benchmark 198

List of Tables

7.1 Combinations of Tags for CIC-indicators "content" 182
7.2 Combinations of Tags for CIC-indicators "level of detail" 182

8.1 Transformers: All Images 192
8.2 Transformers: Pixmap, Greymap, and Bitmap 193
8.3 Combiners 194
8.4 Input/Output 194
8.5 Generators 195
8.6 Query Functions 195
8.7 Images used for the Benchmark 197
8.8 Bitmap Operators 199
8.9 Greymap Operators 1 199
8.10 Greymap Operators 2 199
8.11 Pixmap Operators 1 199
8.12 Pixmap Operators 2 200
8.13 Anymap Operators 1 200
8.14 Anymap Operators 2 200
8.15 Anymap Operators 3 200
8.16 Anymap Operators 4 201
8.17 Anymap Operators 5 201
8.18 Anymap Operators 6 201
8.19 Anymap Operators 7 202
8.20 Anymap Operators 8 202
8.21 Commutativity of Operators for Images and Pixmap Images . . . 208

Bibliography

[Ackermann, 1994] Philipp Ackermann. Direct manipulation of temporal structures in a multimedia application framework. In *Proceedings of the ACM International Conference on Multimedia*, pages 51–58, San Francisco, October 1994. ACM Press.

[ACM, 1993] ACM. *Proceedings of the ACM International Conference on Multimedia*, Anaheim, California, August 1993. ACM Press.

[ACM, 1994] ACM. *Proceedings of the ACM International Conference on Multimedia*, San Francisco, California, October 1994. ACM Press.

[ACM, 1995] ACM. *Proceedings of the ACM International Conference on Multimedia*, San Francisco, California, November 1995. ACM Press.

[Adiba, 1995] Michel Adiba. STORM structural and temporal object-oriented multimedia database systems. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 12–19, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Akscyn *et al.*, 1988] Robert Akscyn, Donald McCracken, and Elise Yoder. KMS: A distributed hypertext system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820–835, 1988.

[Allen, 1983] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Anderson *et al.*, 1994] Kenneth M. Anderson, Richard N. Taylor, and E. James Whitehead, Jr. Chimera: Hypertext for heterogeneous software environments.

In *Proceedings of the ACM Conference on Hypertext*, pages 94–107, Edinburgh, September 1994.

[Andrews *et al.*, 1993] Keith Andrews, Hermann Maurer, and Nick Scherbakov. Semantic modeling in hypermedia: An object-oriented approach. Manuscript, distributed together with HM-Card 1.4, 1993.

[Armbrüster and Wimmer, 1992] Heinrich Armbrüster and Klaus Wimmer. Broadband multimedia applications using ATM networks: High-performance computing, high-capacity storage, and high-speed communication. *IEEE Journal on Selected Areas in Communications*, 10(9):1382–1396, December 1992.

[Arnold *et al.*, 1994a] Stephen C. Arnold, Leo Mark, and Sham Navathe. A model of hypermedia systems for administration of semantic connections. In *Proceedings of the 5th International Hong Kong Computer Society Workshop, Next Generation Database Systems*, pages 155–166, Kowloon Shangri-La Hong Kong, February 1994. Hong Kong Computer Society.

[Arnold *et al.*, 1994b] Stephen C. Arnold, Leo Mark, and Sham Navathe. Specification and efficient monitoring of local graph-based constraints in hypermedia systems. Technical Report GI-CC-94-56, College of Computing, Georgia Institute of Technology, Atlanta, Georgia 30332-0280, November 1994.

[Artale *et al.*, 1996] Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering*, 20(3):347–383, November 1996.

[Athappilly *et al.*, 1994] K. K. Athappilly, Chris Durben, and Steven Woods. Multimedia computing – an overview. In Sorel Reisman, editor, *Multimedia Computing*, chapter 4, pages 103–124. Idea Group Publishing, Harrisburg, USA, London, UK, 1994.

[Bach *et al.*, 1993] Jeffrey R. Bach, Santanu Paul, and Ramesh Jain. A visual information management system for the interactive retrieval of faces. *Knowledge and Data Engineering*, 5(4):619–628, 1993.

[Baeza-Yates and Navarro, 1996] Ricardo Baeza-Yates and Gonzalo Navarro. Integrating contents and structure in text retrieval. *ACM SIGMOD RECORD*, 25(1):67–79, March 1996.

[Balasubramanian *et al.*, 1994] P. Balasubramanian, T. Isakowitz, and E. A. Stohr. Designing hypermedia applications. In Jay F. Nunamaker and Ralph H. Sprague, editors, *Proceedings of the Hawaii International Conference on System Sciences. Volume 3 : Information Systems: DSS/Knowledge-Based Systems*, pages 354–365, Los Alamitos, CA, USA, January 1994. IEEE Computer Society Press.

[Bapat *et al.*, 1996] Ajit Bapat, Jürgen Wäsch, Karl Aberer, and Jörg M. Haake. HyperStorM: An extensible object-oriented hypermedia engine. In *Proceedings of the ACM Conference on Hypertext*, pages 203–214, New York, March 1996. ACM Press.

[Basili *et al.*, 1995] R. Basili, F. Grisoli, and M. T. Pazienza. HIRMA: Hypertextual information retrieval system managed by ARISTO. *Data & Knowledge Engineering*, 17:187–213, 1995.

[Berra *et al.*, 1992] P. B. Berra, C-Y. R. Chen, A. Ghafoor, and T. D. C. Little. Issues in networking and data management of distributed multimedia systems. In *Symposium on High-Performance Distributed Computing*, Syracuse, New York, September 1992.

[Bleisinger *et al.*, 1991] Rainer Bleisinger, Rainer Hoch, and Andreas Dengel. ODA-based modeling for document analysis. Technical Memo TM-91-14, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, Erwin-Schrödinger Strasse, Postfach 2080, 67608 Kaiserslautern, Germany, 1991.

[Blustein, 1997] J. Blustein. Hypertext references. `http://www.csd.uwo.ca/~jamie/.Refs/ht-defn.html`, 1997.

[Boone, 1992] Joan Boone. A survey of data models for hypermedia. Technical report, IBM Corporation and Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175, April 1992.

[Bordegoni, 1992] Monica Bordegoni. Multimedia in views. Technical report, CWI, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, 1992.

[Botafogo *et al.*, 1992] Rodrigo A. Botafogo, Ehud Rivlin, and Ben Shneiderman. Structural analysis of hypertexts: Identifying hierarchies and useful metrics. *ACM Transactions on Information Systems*, 10(2):142–180, April 1992.

[Bra and Houben, 1992] Paul De Bra and Geert-Jan Houben. An extensible data model for hyperdocuments. In *Proceedings of the ACM Conference on Hypertext*, pages 222–231, Milano, Italy, December 1992.

[Breiteneder *et al.*, 1992a] Christian Breiteneder, Laurent Dami, Simon Gibbs, Vicki de Mey, and Dennis Tsichritzis. Telepresence in shared virtual worlds. In D. Tsichritzis, editor, *Object Frameworks*, pages 265–274. Centre Universitaire D'Informatique, Universite De Geneve, 1992.

[Breiteneder *et al.*, 1992b] Christian Breiteneder, Simon Gibbs, and Dennis Tsichritzis. Modeling of audio/video data. In D. Tsichritzis, editor, *Object Frameworks*, pages 293–310. Centre Universitaire D'Informatique, Universite De Geneve, 1992.

[Buchanan and Zellweger, 1992] M. Cecilia Buchanan and Polle T. Zellweger. Specifying temporal behavior in hypermedia documents. In *Proceedings of the ACM Conference on Hypertext*, pages 262–271, Milano, Italy, December 1992.

[Buchanan and Zellweger, 1993] M. Cecilia Buchanan and Polle T. Zellweger. Automatic temporal layout mechanisms. In *Proceedings of the ACM International Conference on Multimedia*, pages 341–350, Anaheim, California, August 1993. ACM Press.

[Bulterman, 1995] Dick C. A. Bulterman. Embedded video in hypermedia documents: supporting integration and adaptive control. *ACM Transactions on Information Systems*, 13(4):440–470, October 1995.

[Bush, 1945] Vannevar Bush. As we may think. *Atlantic Monthly*, pages 101–108, July 1945.

[Campbell and Chung, 1995] Scott T. Campbell and Soon M. Chung. The role of database systems in the management of multimedia information. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 4–11, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Campbell and Goodman, 1988] Brad Campbell and Joseph M. Goodman. HAM: A general-purpose hypertext abstract machine. *Communications of the ACM*, 31(7):856–861, July 1988.

[Cardenas *et al.*, 1993] Alfonso F. Cardenas, Ion Tim Ieong, Ricky K. Tiara, Roger Baker, and Claudine M. Breant. The knowledge-based object-oriented PICQUERY+ language. *Knowledge and Data Engineering*, 5(4):644–657, 1993.

[Cavallaro *et al.*, 1993] Umberto Cavallaro, Franca Garzotto, Paolo Paolini, and Domenico Totaro. HIFI: Hypertext interface for information systems. *IEEE Software*, pages 48–51, November 1993.

[Chaudhuri and Garavano, 1996] Surajit Chaudhuri and Luis Garavano. Optimizing queries over multimedia repositories. In *Proceedings of the ACM SIGMOD Conference on the Management of Data*, pages 91–102, Montreal, Quebec, Canada, June 1996. ACM.

[Christel, 1995] Michael G. Christel. Addressing the contents of video in a digital library. In Isabel F. Cruz, Joe Marks, and Kent Wittenburg, editors, *Electronic Proceedings of the ACM Workshop on Effective Abstractions in Multimedia: Layout, Presentation, and Interaction*, `http://www.cs.tufts.edu/~isabel/christel/christel.html`, San Francisco, California, November 1995. ACM.

[Christophides and Rizk, 1994] V. Christophides and A. Rizk. Querying structured documents with hypertext links using OODBMS. In *Proceedings of the ACM Conference on Hypertext*, pages 186–197, Edinburgh, September 1994.

[Colazzo and Molinari, 1995] L. Colazzo and A. Molinari. To see or not to see: tools for teaching with hypertext slides. In *Proceedings EDMEDIA '95*, pages 157–162. AACE, 1995.

[Conklin and Begeman, 1988] J. Conklin and M. L. Begeman. gIBIS: A hypertext tool for exploratory policy discussion. *ACM Transactions on Information Systems*, 6(4):303–331, 1988.

[Conklin, 1987] J. Conklin. Hypertext: An introduction and survey. *IEEE COMPUTER*, pages 17–41, 1987.

[Dami *et al.*, 1988] L. Dami, E. Fiume, O. Nierstrasz, and D. Tsichritzis. Temporal scripts for objects. In D. Tsichritzis, editor, *Active Object Environments*, pages 144–161. Centre Universitaire D'Informatique, Universite De Geneve, 1988.

[Data & Knowledge Engineering, 1996] Special issue: Modeling parts and wholes. *Data & Knowledge Engineering*, 20(3), November 1996.

[Date, 1990] C. J. Date. *An Introduction to Database Systems*. Systems Programming Series. Addison-Wesley, Reading, Massachusetts, volume 1, 5th edition, 1990.

[Davis and Burke, 1993] Andrew W. Davis and Joe Burke. The Mac goes to the movies: A detailed look at Apple's QuickTime architecture. *BYTE Magazine*, 18(2):225–230, February 1993.

[de Mey and Gibbs, 1993] Vicki de Mey and Simon Gibbs. A multimedia component kit. In *Proceedings of the ACM International Conference on Multimedia*, pages 291–300, Anaheim, California, August 1993. ACM Press.

[de Mey *et al.*, 1992] Vicki de Mey, Christian Breiteneder, Laurent Dami, Simon Gibbs, and Dennis Tsichritzis. Visual composition and multimedia. In D. Tsichritzis, editor, *Object Frameworks*, pages 243–257. Centre Universitaire D'Informatique, Universite De Geneve, 1992.

[Delisle and Schwartz, 1986] Norman Delisle and Mayer Schwartz. Neptune: a hypertext system for CAD applications. In *Proceedings of the ACM SIGMOD Conference on the Management of Data*, pages 132–139, 1986.

[Diaz and Sénac, 1994] Michel Diaz and Patrick Sénac. Time stream Petri nets – a model for timed multimedia information. In *Application and Theory of Petri Nets*, pages 219–238. Springer, 1994.

[Dobrovnik, 1995] Michael Dobrovnik. *Externe Schemata in objektorientierten Datenbanksystemen – Logische Datenunabhängigkeit durch Änderungen über Sichten*. PhD thesis, Universität Klagenfurt, Fakultät für Wirtschaftswissenschaften und Informatik, Klagenfurt, Austria, August 1995.

[Drapeau, 1993] George D. Drapeau. Synchronization in the MAEstro multimedia authoring environment. In *Proceedings of the ACM International Conference on Multimedia*, pages 331–339, Anaheim, California, August 1993. ACM Press.

[Duda and Keramane, 1995] Andrej Duda and Chérif Keramane. Structured temporal composition of multimedia data. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 136–142, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Durand *et al.*, 1994] David Durand, Anja Haake, David Hicks, and Fabio Vitali, editors. *Proceedings of the Workshop On Versioning in Hypertext Systems*, Edinburgh, September 1994.

[Elmasri and Navathe, 1994] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. World Student Series. Benjamin/Cummings, Redwood City, California, 2nd edition, 1994.

[Eun *et al.*, 1993] Seong Bae Eun, Eun Suk No, Hyung Chul Kim, Hyunsoo Yoon, and Seung Ryoul Maeng. Specification of multimedia composition and a visual programming environment. In *Proceedings of the ACM International Conference on Multimedia*, pages 167–173, Anaheim, California, August 1993. ACM Press.

[Fernandez *et al.*, 1997] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for a web-site management system. *ACM SIGMOD RECORD*, 26(3):4–11, September 1997.

[Flickner *et al.*, 1995] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dargutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: The QBIC system. *IEEE COMPUTER*, 28(9):23–32, September 1995.

[Fremdenverkehrszentrale Linz, 1997] Fremdenverkehrszentrale Linz. Linz geht rund – A walk through the old quarter of Linz. Booklet, Urfahrmarkt 1, 4040 Linz, 1997.

[Furht, 1994] Borko Furht. Multimedia systems: An overview. *IEEE Multimedia*, 1(1):47–59, 1994.

[Furuta and Stotts, 1990] Richard Furuta and P. David Stotts. The Trellis hypertext reference model. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 83–93. National Institute of Standards and Technology, January 1990.

[Furuta and Stotts, 1994a] Richard Furuta and P. David Stotts. A hypermedia basis for the specification, documentation, verification, and prototyping of concurrent protocols. Technical Report 94-039, 1994.

[Furuta and Stotts, 1994b] Richard Furuta and P. David Stotts. Trellis: a formally-defined hypertextual basis for integrating task and information. Technical report, 1994.

[Gardarin and Yoon, 1994] Georges Gardarin and Shim Yoon. Modeling and querying structured hypermedia documents. Technical report, PRiSM Laboratory, University of Versailles, December 1994.

[Gardarin and Yoon, 1995] Georges Gardarin and Shim Yoon. Filtering hypermedia network by using view mechanism. In Isabel F. Cruz, Joe Marks, and Kent Wittenburg, editors, *Electronic Proceedings of the ACM Workshop on Effective Abstractions in Multimedia: Layout, Presentation, and Interaction*, `http://www.cs.tufts.edu/~isabel/yoon/yoon.html`, San Francisco, California, November 1995. ACM.

[Garg, 1988] Pankaj K. Garg. Abstraction mechanisms in hypertext. *Communications of the ACM*, 31(7):862–870, July 1988.

[Garzotto and Paolini, 1991] Franca Garzotto and Paolo Paolini. HDM - a model for the design of hypertext applications. In *Proceedings of the ACM Conference on Hypertext*, pages 313–328, 1991.

[Garzotto *et al.*, 1993a] F. Garzotto, L. Mainetti, and P. Paolini. Navigation in hypermedia applications: Modelling and semantics. Technical Report 93-034, Dipartimento di Elettronica, Politecnico di Milano, May 1993.

[Garzotto *et al.*, 1993b] Franca Garzotto, Luca Mainetti, and Paolo Paolini. HDM2: Extending the E-R approach to hypermedia application design. In *Proceedings of the International Conference on the Entity-Relationship Approach*, pages 178–189. Springer, 1993.

[Garzotto *et al.*, 1993c] Franca Garzotto, Luca Mainetti, and Paolo Paolini. Structured design of hypermedia applications. In *Proceedings of the International Workshop on Hypermedia Interfaces*, 1993.

[Garzotto *et al.*, 1993d] Franca Garzotto, Paolo Paolini, and Daniel Schwabe. HDM - a model-based approach to hypertext application design. *ACM Transactions on Information Systems*, 11(1):1–26, January 1993.

[Garzotto *et al.*, 1994] Franca Garzotto, Luca Mainetti, and Paolo Paolini. Adding multimedia collections to the Dexter model. In *Proceedings of the ACM Conference on Hypertext*, pages 70–80, Edinburgh, September 1994.

[Gemmell and Christodoulakis, 1992] J. Gemmell and S. Christodoulakis. Principles of delay sensitive multimedia data storage and retrieval. *ACM Transactions on Information Systems*, 10(1):51–90, January 1992.

[Gemmell, 1995] D. James Gemmell. Multimedia storage servers: A tutorial. *IEEE COMPUTER*, 28(5):40–49, September 1995.

[Ghafoor, 1995] Arif Ghafoor. Multimedia database management systems. *ACM Computing Surveys*, 27(4):593–598, December 1995.

[Gibbs and Tsichritzis, 1995] Simon J. Gibbs and Dionysios C. Tsichritzis. *Multimedia Programming – Objects, Environments and Frameworks.* Addison-Wesley, Reading, Massachusetts, 1995.

[Gibbs *et al.*, 1992] Simon Gibbs, Christian Breiteneder, Laurent Dami, Vicki de Mey, and Dennis Tsichritzis. A programming environment for multimedia applications. In D. Tsichritzis, editor, *Object Frameworks*, pages 259–264. Centre Universitaire D'Informatique, Universite De Geneve, 1992.

[Gibbs *et al.*, 1993a] Simon Gibbs, Christian Breiteneder, Vicki de Mey, and Dennis Tsichritzis. Video widgets and video actors. In *Proceedings UIST '92*, 1993.

[Gibbs *et al.*, 1993b] Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. Audio/video databases: An object-oriented approach. In *Proceedings of the 9th International Conference on Data Engineering*, pages 381–390, 1993.

[Gibbs *et al.*, 1994] Simon Gibbs, Christian Breiteneder, and Dennis Tsichritzis. Data modeling of time-based media. In *Proceedings of the ACM SIGMOD Conference on the Management of Data*, pages 91–102. ACM, May 1994.

[Gibbs, 1991] Simon Gibbs. Composite multimedia and active objects. In *Proceedings OOPSLA '91*, pages 97–112, 1991.

[Grønbæk and Trigg, 1994] Kaj Grønbæk and Randall H. Trigg. Design issues for a Dexter-based hypermedia system. *Communications of the ACM*, 37(2):40–49, 1994.

[Grønbæk *et al.*, 1993] Kaj Grønbæk, Jens A. Hem, Ole L. Madsen, and Lennert Sloth. Designing Dexter-based cooperative hypermedia systems. In *Proceedings of the ACM Conference on Hypertext*, pages 25–38. ACM Press, March 1993.

[Grønbæk *et al.*, 1994] Kaj Grønbæk, Jens A. Hem, Ole L. Madsen, and Lennert Sloth. Cooperative hypermedia systems: a Dexter-based architecture. *Communications of the ACM*, 37(2):64–74, February 1994.

[Grønbæk, 1994] Kaj Grønbæk. Composites in a Dexter-based hypermedia framework. In *Proceedings of the ACM Conference on Hypertext*, pages 59–69, Edinburgh, September 1994.

[Grosky, 1997] William I. Grosky. Managing multimedia – information in database systems. *Communications of the ACM*, 40(12):73–80, December 1997.

[Gu and Neuhold, 1993] Junzhong Gu and Erich J. Neuhold. A data model for multimedia information retrieval. In *Proceedings of the First International Conference on Multi-Media Modeling*, pages 113–127. World Scientific, Singapure, November 1993.

[Guarino *et al.*, 1996] Nicola Guarino, Simone Pribbenow, and Laure Vieu. Modeling parts and wholes. *Data & Knowledge Engineering*, 20(3):257–258, November 1996.

[Gudivada and Raghavan, 1995] Venkat N. Gudivada and Vijay V. Raghavan. Content-based image retrieval systems. *IEEE COMPUTER*, 28(9):18–22, September 1995.

[Guglielmo and Rowe, 1996] Eugene J. Guglielmo and Neil C. Rowe. Natural-language retrieval of images based on descriptive captions. *ACM Transactions on Information Systems*, 14(3):237–267, July 1996.

[Gupta *et al.*, 1991] Amarnath Gupta, Terry Weymouth, and Ramesh Jain. Semantic queries with pictures: The VIMSYS model. In Guy M. Lohman, Amilcar Sernadas, and Rafael Camps, editors, *Proceedings of the International Conference on Very Large Databases*, pages 69–79, Barcelona, Spain, September 1991. Morgan Kaufmann.

[Haake, 1992] Anja Haake. CoVer: A contextual version server for hypertext applications. In D. Lucarella, J. Nanard, M. Nanard, and P. Paolini, editors, *Proceedings of the ACM Conference on Hypertext*, Versioning, pages 43–52, Milano, Italy, November 1992. ACM Press.

[Halasz and Schwartz, 1990] Frank Halasz and Mayer Schwartz. The Dexter hypertext reference model. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 95–133. National Institute of Standards and Technology, January 1990.

[Halasz and Schwartz, 1994] Frank Halasz and Mayer Schwartz. The Dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

[Halasz, 1988] F. Halasz. Reflections on NoteCards: Seven issues for the next generation of hypertext systems. *Communications of the ACM*, 31(7):836–851, 1988.

[Halper *et al.*, 1992] Michael Halper, James Geller, and Yehoshua Perl. "Part" relations for object-oriented databases. In G. Pernul and A. M. Tjoa, editors, *Proceedings of the International Conference on the Entity-Relationship Approach*, pages 406–422, Karlsruhe, Germany, October 1992. Springer.

[Hamakawa and Rekimoto, 1994] Rei Hamakawa and Jun Rekimoto. Object composition and playback models for handling multimedia data. *Multimedia Systems*, 2(1):26–35, 1994.

[Harney *et al.*, 1991] Kevin Harney, Mike Keith, Gary Lavelle, Lawrence D. Ryan, and Daniel J. Stark. The i750 video processor: A total multimedia solution. *Communications of the ACM*, 34(4):64–78, April 1991.

[Hull and King, 1987] Richard Hull and Roger King. Semantic database modeling: Survey, applications, and research issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[IBM Corporation, 1994] IBM Corporation. Multimedia distributed computing. In Sorel Reisman, editor, *Multimedia Computing*, chapter 2, pages 34–84. Idea Group Publishing, Harrisburg, USA, London, UK, 1994.

[Illustra Information Technologies, Inc., 1994a] Illustra Information Technologies, Inc. *Illustra User's Guide*, June 1994. Illustra Server Release 2.1.

[Illustra Information Technologies, Inc., 1994b] Illustra Information Technologies, Inc. *Image DataBlade Guide*, June 1994. Illustra Image DataBlade Release 1.2.

[Illustra Information Technologies, Inc., 1994c] Illustra Information Technologies, Inc. *System Administrator's Guide*, June 1994. Illustra Server Release 2.1.

[Kacmar and Leggett, 1991] Charles J. Kacmar and John J. Leggett. PROXHY: A process-oriented extensible hypertext architecture. *ACM Transactions on Information Systems*, 9(4):399–419, 1991.

[Keim and Lum, 1993] Daniel A. Keim and Vincent Lum. GRADI: A graphical database interface for a multimedia DBMS. In *Proceedings International Workshop on Interfaces to Database Systems 1992*, pages 95–112, Glasgow, 1993. Workshops in Computing, Springer.

[Kimber, 1993] W. Eliot Kimber. HyTime and SGML: Understanding the HyTime HyQ query language. Technical report, August 1993. `ftp://ftp.ifi.uio.no/pub/SGML/HyTime`.

[Koegel *et al.*, 1993] John F. Koegel, Lloyd W. Rutledge, John L. Rutledge, and Can Keskin. HyOctane: A HyTime engine for an MMIS. In *Proceedings of* *the ACM International Conference on Multimedia*, pages 129–136, Anaheim, California, August 1993. ACM Press.

[Kwon and Lee, 1995] Taeck-Geun Kwon and Sukho Lee. Data placement for continuous media in multimedia DBMS. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 110–117, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Lange, 1990] Danny B. Lange. A formal model of hypertext. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 145–166. National Institute of Standards and Technology, January 1990.

[Lange, 1994] Danny B. Lange. An object-oriented design method for hypermedia information systems. In Jay F. Nunamaker and Ralph H. Sprague, editors, *Proceedings of the Hawaii International Conference on System Sciences. Volume 3: Information Systems: DSS/Knowledge-Based Systems*, pages 366–375. IEEE Computer Society Press, January 1994.

[Layaïda and Keramane, 1995] Nabil Layaïda and Chérif Keramane. Maintaining temporal consistency of multimedia documents. In Isabel F. Cruz, Joe Marks, and Kent Wittenburg, editors, *Electronic Proceedings of the ACM Workshop on Effective Abstractions in Multimedia: Layout, Presentation, and Interaction*, `http://www.cs.tufts.edu/~isabel/Layaida/ACM95-1.html`, San Francisco, California, November 1995. ACM.

[Leggett and Schnase, 1994] John J. Leggett and John L. Schnase. Viewing Dexter with open eyes. *Communications of the ACM*, 37(2):76–86, February 1994.

[Lenssen, 1994] Klaus Lenssen. Enabling hypermedia messaging BPRM — a body part relationship module. In *Proceedings of the ACM International Conference on Multimedia*, pages 157–163, San Francisco, California, October 1994. ACM Press.

[Leung and Zheng, 1995] C. H. C. Leung and Z. J. Zheng. Image data modelling for efficient content indexing. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 143–150, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Liebowitz and Bland, 1994] Jay Liebowitz and Karen Bland. Using multimedia to help students learn knowledge acquisition. In Sorel Reisman, editor, *Multimedia Computing*, chapter 17, pages 536–557. Idea Group Publishing, Harrisburg, USA, London, UK, 1994.

[Lindblad *et al.*, 1994] Christopher J. Lindblad, David J. Wetherall, and David L. Tennenhouse. The VuSystem: A programming system for visual processing of digital video. In *Proceedings of the ACM International Conference on Multimedia*, pages 307–314, San Francisco, California, October 1994. ACM Press.

[Lisberger, 1998] Robert Lisberger. Implementierung eines Prototyps einer Hypermedia Datenbank Architektur für Bilder und Texte. Master's thesis, Johannes Kepler Universität Linz, Institut für Wirtschaftsinformatik, Datenbanken und Wissensbasierte Systeme, 1998. To be submitted in May 1998.

[Little and Ghafoor, 1993] T. D. C. Little and A. Ghafoor. Interval-based conceptual models for time-dependent multimedia data. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):551–563, August 1993.

[Marshall, 1990] Catherine C. Marshall. A multi-tiered approach to hypertext integration: Negotiating standards for a heterogeneous application environment. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 167–177. National Institute of Standards and Technology, January 1990.

[Maurer *et al.*, 1993a] Hermann Maurer, F. Kappe, Nick Scherbakov, and P. Srinivasan. Structured browsing of hypermedia databases. In *Proceedings Human Computer Interaction*, pages 51–62. Lecture Notes in Computer Science 733, Springer, 1993.

[Maurer *et al.*, 1993b] Hermann Maurer, Nick Scherbakov, and P. Srinivasan. A new hypermedia data model. In *Proceedings of the International Conference on Database and Expert Systems Applications*, pages 685–696, Prague, Czech Republic, September 1993. Springer.

[Maurer *et al.*, 1994] Hermann Maurer, Keith Andrews, and Nick Scherbakov. Browsing hypermedia composites: An algebraic approach. Submitted to ACM Transactions on Information Systems, 1994.

[Mayfield and Nicholas, 1993] James Mayfield and Charles Nicholas. SNITCH: Augmenting hypertext documents with a semantic net. *International Journal of Intelligent and Cooperative Information Systems*, 2(3):335–351, 1993.

[Mayfield, 1995] James Mayfield. Two-level models of hypertext. In R. D. Semmel, editor, *Advances in Software Engineering and Knowledge Engineering*, Software Engineering and Knowledge Engineering. World Publishing, 1995.

[McHugh *et al.*, 1997] Jason McHugh, Serge Abiteboul, Roy Goldman, Dallan Quass, and Jennifer Widom. Lore: A database management system for semistructured data. *ACM SIGMOD RECORD*, 26(3):54–66, September 1997.

[Meyer-Boudnik and Effelsberg, 1995] Thomas Meyer-Boudnik and Wolfgang Effelsberg. MHEG explained. *IEEE Multimedia*, 2(1):26–38, Spring 1995.

[Nahrstedt, 1995] Klara Nahrstedt. Resource management in networked multimedia systems. *IEEE COMPUTER*, 28(5):52–63, September 1995.

[Narasimhalu, 1996] A. Desai Narasimhalu. Multimedia databases. *Multimedia Systems*, 4(5):226–249, 1996.

[Narayana *et al.*, 1995] Sandhya Narayana, Paul Losleben, and Fah-Chun Cheong. A market-based economic model for multi-media object storage and distribution. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 118–126, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Nelson, 1965] Theodor Holm Nelson. A file structure for the complex, the changing and the indeterminate. In *Proceedings of the ACM 20th National Conference*, pages 84–100, 1965.

[Nelson, 1967] Theodor Holm Nelson. Getting out of our system. In G. Schechter, editor, *Information Retrieval: A Critical Review*, pages 191–210, Washington, D.C., 1967. Thompson Books.

[Nelson, 1990] Theodor Holm Nelson. *Literary Machines: the report on, and of, Project Xanadu concerning word processing, electronic publishing, hypertext, thinkertoys tomorrow's intellectual revolution, and certain other topics including knowledge, education and freedom.* Mindfull Press, Sausalito, Canada, 90.1 edition, 1990.

[Newcomb *et al.*, 1991] Steven R. Newcomb, Neill A. Kipp, and Victoria T. Newcomb. The "HyTime" hypermedia/time-based document structuring language. *Communications of the ACM*, 34(11):67–83, November 1991.

[Newcomb, 1995] S. R. Newcomb. Multimedia interchange using SGML/HyTime. Part I: Structures. *IEEE Multimedia*, 2(2):86–89, Summer 1995.

[Object Design, Inc., 1997] Object Design, Inc. Object designs: Objectstore object managers. `http://www.odi.com/Products/ObjMgr_Homepage.html`, 1997.

[O'Connell *et al.*, 1996] W. O'Connell, I. T. Ieong, D. Schrader, C. Watson, G. Au, A. Biliris, S. Choo, P. Colin, G. Linderman, E. Panagos, J. Wang, and T. Walter. A teradata content-based multimedia object manager for massively parallel architectures. In *Proceedings of the ACM SIGMOD Conference on the Management of Data*, pages 68–78, Montreal, Quebec, Canada, June 1996. ACM Press.

[Ogle and Stonebraker, 1995] Virginia E. Ogle and Michael Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE COMPUTER*, 28(9):40–48, September 1995.

[Oomoto and Tanaka, 1993] Etetsu Oomoto and Katsumi Tanaka. OVID: Design and implementation of a video-object database system. *Knowledge and Data Engineering*, 5(4):629–643, 1993.

[Oracle Corporation, 1997] Oracle Corporation. Websight data cartridges. `http://www.oracle.com/st/cartridges/index.html`, 1997.

[Østerbye and Nørmark, 1994] Kasper Østerbye and Kurt Nørmark. An interaction engine for rich hypertexts. In *Proceedings of the ACM Conference on Hypertext*, pages 167–176, Edinburgh, September 1994.

[Østerbye, 1992] Kasper Østerbye. Structural and cognitive problems in providing version control for hypertext. In *Proceedings of the ACM Conference on Hypertext*, Versioning, pages 33–42, November 1992.

[Ozden *et al.*, 1996] Banu Ozden, Rajeev Rastogi, Prashant Shenoy, and Avi Silberschatz. Fault-tolerant architectures for continuous media servers. In *Proceedings of the ACM SIGMOD Conference on the Management of Data*, pages 79–90, Montreal, Quebec, Canada, June 1996. ACM Press.

[Parunak, 1990a] H. Van Dyke Parunak. Reference and data model group (RDMG): Work plan status. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 9–13. National Institute of Standards and Technology, January 1990.

[Parunak, 1990b] H. Van Dyke Parunak. Toward a reference model for hypermedia. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 197–211. National Institute of Standards and Technology, January 1990.

[Picard, 1995] Rosalind W. Picard. Computer learning of subjectivity. *ACM Computing Surveys*, 27(4):621–623, December 1995.

[Pillwein, 1824] Benedikt Pillwein. *Geschichte der Stadt Linz*. Verlag Wilhelm Ennsthaler, Steyr, Austria, 1824. Replica, printed 1966.

[Prabhakaran and Raghavan, 1993] B. Prabhakaran and S. V. Raghavan. Synchronization models for multimedia presentations with user participation. In *Proceedings of the ACM International Conference on Multimedia*, pages 157–166, Anaheim, California, August 1993. ACM Press.

[Price, 1993] Roger Price. MHEG: An introduction to the future international standard for hypermedia object interchange. In *Proceedings of the ACM International Conference on Multimedia*, pages 121–128, Anaheim, California, August 1993. ACM Press.

[Prückler and Schrefl, 1995] Thomas Prückler and Michael Schrefl. Modeling corresponding information content between multimedia data. In S. Fraisse, F. Garzotto, T. Isakowitz, J. Nanard, and M. Nanard, editors, *Hypermedia Design:*

Proceedings of the International Workshop on Hypermedia Design (IWHD'95), pages 59–69, Montpellier, France, June 1995. Workshops in Computing, Springer.

[Prückler and Schrefl, 1996] Thomas Prückler and Michael Schrefl. Achieving physical data independence in hypermedia databases. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 141–150, Blue Mountain Lake, New York, August 1996. IEEE Computer Society Press.

[Prückler and Schrefl, 1997a] Thomas Prückler and Michael Schrefl. Achieving logical data independence in hypermedia databases. In R. Wagner, editor, *Proceedings of the 8th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 160–169, Toulouse, France, September 1997. IEEE Computer Society.

[Prückler and Schrefl, 1997b] Thomas Prückler and Michael Schrefl. Whole-part relationships in hypermedia databases. In *Proceedings International Conference on Multimedia Information Systems (in conjunction with International Conference on Advanced Science and Technology)*, pages 68–78, Schaumburg, Illinois, April 1997.

[Qazi *et al.*, 1993] Naveed U. Qazi, Miae Woo, and Arif Ghafoor. A synchronization and communication model for distributed multimedia objects. In *Proceedings of the ACM International Conference on Multimedia*, pages 147–156, Anaheim, California, August 1993. ACM Press.

[Raymond and Tompa, 1988] Darrell R. Raymond and Frank Wm. Tompa. Hypertext and the oxford english dictionary. *Communications of the ACM*, 31(7):871–879, July 1988.

[Riley, 1990] Victor A. Riley. An interchange format for hypertext systems: The intermedia model. In Judi Moline, Dan Benigni, and Jean Baronas, editors, *Proceedings of the Hypertext Standardization Workshop*, pages 213–222. National Institute of Standards and Technology, January 1990.

[Robertson *et al.*, 1994] John Robertson, Erik Merkus, and Athula Ginige. The hypermedia authoring research toolkit (HART). In *Proceedings of the ACM Conference on Hypertext*, pages 177–185, Edinburgh, September 1994.

[Rumbaugh *et al.*, 1991] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, 1991.

[Rumbaugh, 1993] James Rumbaugh. On the horns of the modeling dilemma – choosing among alternate modeling constructs. *Journal of Object-Oriented Programming*, 6(7):8–17, 1993.

[Rundensteiner, 1992] Elke A. Rundensteiner. MultiView: A methodology for supporting multiple views in object-oriented databases. In Li-Yan Yuan, editor, *Proceedings of the International Conference on Very Large Databases*, pages 187–198, Vancouver, Canada, August 1992. Morgan Kaufmann.

[Schloss and Wynblatt, 1995] Gerhard A. Schloss and Michael J. Wynblatt. Building temporal structures in a layered multimedia data model. In *Proceedings of the ACM International Conference on Multimedia*, pages 271–278. ACM Press, October 1995.

[Schnase *et al.*, 1993] John L. Schnase, John J. Leggett, David L. Hicks, and Ron L. Szabo. Semantic data modeling of hypermedia associations. *ACM Transactions on Information Systems*, 11(1):27–50, 1993.

[Scholl *et al.*, 1991] Marc H. Scholl, Christian Laasch, and Markus Tresch. Updatable views in object-oriented databases. In C. Delobel, M. Kifer, and Y. Masunaga, editors, *Proceedings of the International Conference on Deductive and Object-Oriented Databases*, pages 189–207, Munich, Germany, December 1991. Springer.

[Schwabe *et al.*, 1996] D. Schwabe, G. Rossi, and S. D. J. Barbosa. Systematic hypermedia application design with OOHDM. In *Proceedings of the ACM Conference on Hypertext*, pages 116–128, Washington, DC, March 1996. ACM Press.

[Shackelford *et al.*, 1993] Douglas E. Shackelford, John B. Smith, and F. Donelson Smith. The architecture and implementation of a distributed hypermedia storage system. In *Proceedings of the ACM Conference on Hypertext*, pages 1–13. ACM Press, March 1993.

[Sheck and Scholl, 1984] H. J. Sheck and M. Scholl. An algebra for the relational model with relation valued attributes. Technical Report TR DVSI-1984-TI, Technical University of Darmstadt, Darmstadt, Germany, 1984.

[Silberschatz *et al.*, 1996] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts*. Addison-Wesley, Reading, Massachusetts, 3rd edition, 1996.

[Simonnot and Smaïl, 1995] Brigitte Simonnot and Malika Smaïl. Model for interactive retrieval of videos and still images. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 128–135, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Smith and Weiss, 1988] John B. Smith and Stephen F. Weiss. Hypertext. *Communications of the ACM*, 31(7):816–819, July 1988.

[Smith, 1996] Barry Smith. Mereotopology: A theory of parts and boundaries. *Data & Knowledge Engineering*, 20(3):287–303, November 1996.

[Speegle, 1995] Greg Speegle. Views of media objects in multimedia databases. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 20–27, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Srihari, 1995] Rohini K. Srihari. Automatic indexing and content-based retrieval of captioned images. *IEEE COMPUTER*, 28(9):49–56, September 1995.

[Staehli *et al.*, 1995] Richard Staehli, Jonathan Walpole, and David Maier. Device and physical data independence for multimedia presentations. *ACM Computing Surveys*, 27(4):640–642, December 1995.

[Stallman, 1995] Richard Stallman. *GNU Emacs manual*. Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA, 11th edition, Emacs version 19.29, June 1995.

[Steinmetz, 1993] Ralf Steinmetz. *Multimedia-Technologie – Einführung und Grundlagen.* Springer, Berlin, Heidelberg, New York, 1993.

[Stonebraker and Kemnitz, 1991] Michael Stonebraker and Greg Kemnitz. The POSTGRES next-generation database management system. *Communications of the ACM*, 34(10):78–93, October 1991.

[Stotts and Furuta, 1989] P. Davis Stotts and Richard Furuta. Petri-net-based hypertext: Document structure with browsing semantics. *ACM Transactions on Information Systems*, 7(1):3–29, 1989.

[Thimm and Klas, 1995] Heiko Thimm and Wolfgang Klas. Playout management – an integrated service of a multimedia database management system. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 38–47, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Tichy, 1985] Walter F. Tichy. RCS: A system for version control. *Software Practice and Experience*, 15(7):637–654, July 1985.

[Tompa, 1989] Frank Wm. Tompa. A data model for flexible hypertext database systems. *ACM Transactions on Information Systems*, 7(1):85–100, January 1989.

[Tsichritzis and Klug, 1978] D. C. Tsichritzis and A. Klug. The ANSI/X3/-SPARC DBMS framework report of the study group on database management systems. *Information Systems*, 3(3):173–191, 1978.

[Ullman, 1988] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, Rockville, Maryland, 1988.

[Utting and Yankelovich, 1989] Kenneth Utting and Nicole Yankelovich. Context and orientation in hypermedia networks. *ACM Transactions on Information Systems*, 7(1):58–84, 1989.

[van Rossum *et al.*, 1993] Guido van Rossum, Jack Jansen, K. Sjoerd Mullender, and Dick C. A. Bulterman. CMIFed: A presentation environment for portable hypermedia documents. In *Proceedings of the ACM International Conference on Multimedia*, pages 183–188, Anaheim, California, August 1993. ACM Press.

[Varzi, 1996] Achille C. Varzi. Parts, wholes, and part-whole relations: The prospects of mereotopology. *Data & Knowledge Engineering*, 20(3):259–286, November 1996.

[Vazirgiannis and Hatzopoulos, 1993] Michael Vazirgiannis and Michael Hatzopoulos. A script based approach for interactive multimedia applications. In *Proceedings of the First International Conference on Multi-Media Modeling*, pages 129–141. World Scientific, Singapure, November 1993.

[Vazirgiannis and Hatzopoulos, 1995] Michael Vazirgiannis and Michael Hatzopoulos. Integrated multimedia object and application modelling based on events and scenarios. In *Proceedings of the International Workshop on Multi-Media Database Management Systems*, pages 48–55, Blue Mountain Lake, New York, August 1995. IEEE Computer Society Press.

[Watters and Shepherd, 1990] Carolyn Watters and Michael A. Shepherd. A transient hypergraph-based model for data access. *ACM Transactions on Information Systems*, 8(2):77–102, April 1990.

[Wiil and Leggett, 1997] Uffe K. Wiil and John J. Leggett. Hyperform : a hypermedia system development environment. *ACM Transactions on Information Systems*, 15(1):1–31, January 1997.

[Wiil, 1991] U. K. Wiil. Issues in the design of EHTS: a multiuser hypertext system for collaboration. In V. Milutinovic, B. D. Shriver, J. F. Nunamaker, Jr., and R. H. Sprague, Jr., editors, *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences (Cat. No. 91TH0394-7)*, pages 629–639 (vol. 2), 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1991. IEEE Computer Society Press.

[Wiil, 1993] Uffe Kock Wiil. Experiences with HyperBase: A hypertext database supporting collaborative work. *ACM SIGMOD RECORD*, 22(4):19–25, December 1993.

[Woelk *et al.*, 1986] Darrell Woelk, Won Kim, and Willis Luther. An object-oriented approach to multimedia databases. In *Proceedings of the ACM SIGMOD Conference on the Management of Data*, pages 311–325, May 1986.

[Woodhead, 1991] Nigel Woodhead. *Hypertext and Hypermedia – Theory and Applications*. Addison-Wesley, Reading, Massachusetts, 1991.

[Wu and Narasimhalu, 1994] Jian Kang Wu and Arcot Desai Narasimhalu. Identifying faces using multiple retrievals. *IEEE Multimedia*, 1(2):27–38, 1994.

[Wu *et al.*, 1994] J.K. Wu, Y.H. Ang, P. Lam, H.H. Loh, and A. Desai Narasimhalu. Inference and retrieval of facial images. *Multimedia Systems*, 2(1):1–14, 1994.

[Yankelovich *et al.*, 1988] Nicole Yankelovich, Bernard J. Haan, Norman K. Meyrowitz, and Steven M. Drucker. Intermedia: The concept and construction of a seamless information environment. *IEEE COMPUTER*, 21(1):81–96, January 1988.

Curriculum Vitae

Personal Record

Name:	Mag. Thomas Prückler
Date of Birth:	31 January, 1968
Place of Birth:	Linz, Upper Austria
Citizenship:	Austria
Home Address:	Landwiedstraße 11
	A-4020 Linz, Austria

University Education

1987 - 1993	Study of Business Information Systems at the Johannes Kepler University of Linz, Austria
1993 - 1998	Ph.D. study at the Johannes Kepler University of Linz, Austria

Professional Experience

- **Fall term 1990/91:** Tutor for courses in computer algorithms at the Department of Business Information Systems, Johannes Kepler University of Linz

- **1992 - 1993:** Student assistant ("Studienassistent") at the Department of Business Information Systems, Data & Knowledge Engineering, Johannes Kepler University of Linz, headed by Prof. Michael Schrefl

- **Fall term 1992/93:** Tutor for courses in database management systems at the Department of Business Information Systems, Johannes Kepler University of Linz

- **1993 - 1997:** Teaching and research associate ("Universitätsassistent") at the Department of Business Information Systems, Data & Knowledge Engineering, Johannes Kepler University of Linz, headed by Prof. Michael Schrefl