CONTENT AND MULTIMEDIA DATABASE MANAGEMENT SYSTEMS

ARJEN P. DE VRIES
Centre for Telematics and Information Technology
University of Twente
The Netherlands
arjen@acm.org

Samenstelling van de promotiecommissie:

Prof. dr. P.M.G. Apers, promotor
Prof. C.J. van Rijsbergen, University of Glasgow, Glasgow, UK
Prof. dr. M.L. Kersten, Universiteit van Amsterdam
Prof. dr. F.M.G. de Jong
Prof. dr. W. Jonker

dr. H.M. Blanken (assistent-promotor)
dr. G.C. van der Veer, Vrije Universiteit, Amsterdam (assistent-promotor)
dr. A.N. Wilschut (referent)

Centre for Telematics and Information Technology (CTIT)
P.O. Box 217, 7500 AE Enschede, The Netherlands

ISBN: 90-365-1388-X
ISSN: 1381-3617 (CTIT Ph.D-thesis Series No. 99-26)

Cover design: Willem G. Feijen
Printed by: PrintPartners Ipskamp, Enschede, The Netherlands

CONTENT AND MULTIMEDIA DATABASE MANAGEMENT SYSTEMS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. F. A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 17 december 1999 te 15.00 uur.

door

Arjen Paul de Vries

geboren op 18 september 1972
te Laren Noord-Holland

Dit proefschrift is goedgekeurd door:

Prof. dr. P.M.G. Apers (promotor)
Dr. H.M. Blanken (assistent-promotor)
Dr. G.C. van der Veer (assistent-promotor)

Contents

Preface

Multimedia is a sexy topic. But, why is it so fascinating? And, if it is so fascinating, why does it seem as if the multimedia hype at database conferences (that started a couple of years ago) has suddenly passed by? The latter may be understood because a business model for commercial applications of multimedia database technology has not yet evolved. But, in words of Albert Camus: 'a society based on production is only productive, not creative'.

To me, creation is the most human activity that exists: I believe it is our main source to happiness. I think it is the direct relationship between multimedia and the creativity necessary for the production of multimedia that is so fascinating. As such, multimedia relates clearly to Art: photographs, paintings, videoclips, movies, songs, etcetera. Obviously, not all multimedia is Art. Conversely, most multimedia is just a representation of Reality. And, Jeanette Winterson argues: 'Art does not imitate life. Art anticipates life.' In other words, true Art is more than a representation of Reality; it forces *you* to enjoy the Artwork itself.

Perceiving Art thus emphasizes the role of Emotions and Aesthetics. But, *whenever* we look at a picture, listen to music, or 'just' watch a news fragment on CNN, we cannot avoid judging also the aesthetics of the scenes perceived, rating them unconsciously by their artistic value. As a result, multimedia data has infinite semantics: every individual has his or her own private perception. This makes the individual user an important factor in the design of multimedia database systems; motivating this thesis's emphasis on the role of aesthetics and emotional value in the minds of individuals when perceiving multimedia. The human factor, adding to the technical challenge of creating large multimedia database systems, has motivated me these four years, and motivates me still.

ARJEN P. DE VRIES

'Eigenlijk zijn alleen muziek en abstracte schilderkunst volledig zelfstandige kunstvormen. Omdat het voor de maker onmogelijk is om het over de werkelijkheid te hebben, ben je als kijker of luisteraar gedwongen van het kunstwerk zelf te genieten. Wat jij doet, over een onweer vertellen en proberen dat zo goed mogelijk te beschrijven...'
'...het gevoel van onweer,' zei ik.
'Dat is hetzelfde. Wat jij doet is een zwakke poging om een nieuwe werkelijkheid te maken, zonder dat je ooit loskomt van de werkelijkheid waaraan je je beelden ontleent.'
—Marcel Möring, *In Babylon*

Acknowledgments

Rolf de By made me first consider the idea of becoming a graduate student, and T.V. Raman convinced me to really do so; a decision I will never regret. The Centre for Telematics and Information Technology (CTIT) provided funding for an inter-disciplinary research project between ergonomics and databases, which resulted in my project. Unfortunately, Twente's cozy little cognitive ergonomics group has fallen apart; but, I am sure you all notice your impact throughout my dissertation, and I remember kindly the production of the Gaze video. Lucky for me, the database group has been a stable second home. The informal atmosphere has been very pleasant to work in, and I thank all my colleagues for their great company. A special word of thanks goes to David Spelt, with whom I shared the full experience of being a graduate student: both the rough times (we disagree on this, but David will put 'seeing Chinese box' here) and the excellent times (we agree on this: Capri).

I want to thank my promotor Peter Apers for providing an environment in which I could choose my own research directions and teach only what I wanted to, and for agreeing with all those trips abroad and stimulating me to do a summer internship with Digital. Peter and my assistent-promotors Gerrit van der Veer and Henk Blanken must have suffered heavily (and frequently) under the numerous papers and ideas I wanted to address in my research. Thanks for never stopping me, giving me complete freedom, and listening patiently to my rattling on about yet another 'great idea'. I am honoured that Keith van Rijsbergen, Martin Kersten, Franciska de Jong, Wim Jonker, and Annita Wilschut agreed kindly to $\bowtie$ my committee.

Without the help of Peter Boncz and the Monet team, my research would have been infeasible. More than anyone else, Annita Wilschut has taught me the essence of database technology; and, a great deal about life as well. Together with Jan Flokstra, she created the framework that is presented in Chapter 2 as the So-Simple DBMS. Maurice van Keulen has been very helpful with proofreading, especially with Chapter 2. Also, I believe that his timely return to the database group has brought both of us a deeper understanding of Moa. Mark van Doorn, Erik van het Hof, Henk Ernst Blok, and Harold Oortwijn contributed significantly to the development of the Mirror DBMS. Finally, Dick Theissens of Symbol Automatisering encouraged experiments with music retrieval, and provided data for the music retrieval experiments. Willem

Feijen designed the beautiful artwork for the cover, demonstrating not only his skill, but also an apt interpretation of the line of reasoning presented in Chapter 3.

International contacts added a lot of fun to doing research. Brian Eberman arranged an awesome summer for me at Digital's Cambridge Research Lab (CRL), where I teamed up with Rosie, Beth, and Oren. The turbulent year following this summer has affected me more deeply than I can describe here. As a result from our meeting at IRSG, Marjo Markkula invited me to an informal Mira workgroup meeting in Tampere, and Keith van Rijsbergen invited me to Glasgow to give a talk to his group; these events made me realize that I could really contribute to IR research. Finally, David McG. Squire has made it possible for me to come to Genève to discuss our remarkably similar research interests. But, travelling has never been necessary to meet interesting people. Right here in Twente, Paul van der Vet has invited me to teach in his information retrieval course; the cooperation with Djoerd Hiemstra for our participation in TREC has been perfect. Also, the informal meetings organized under the MMUIS banner have been an excellent platform to gain self-confidence. At SIKS courses, I found kindred spirits in Bastiaan, Martijn, and Inge.

Which brings me to my friends and family, who I have seriously neglected during these four years; blaming this on lack of time on the one hand (especially during this final year), and geographical location on the other hand (Enschede is simply too far away). You should realize that your continued support has been very important, without which I would never have finished. My friends from Euros and Tibagem have always provided the best distractions from my addiction to work. And for my roommates, Mariken, Karen, and Vlora, I can only hope our fun times at the Dommelstraat have outweighed all those times I skipped dinner, whether I was going abroad or just living in my office instead. Wim and Ria, thanks for your support and care. Papa, mama, and Milou, thanks for believing in me always. And Kristel, thanks for showing me that together we can survive everything ... even four crazy years like these.

To Kristel

1
INTRODUCTION

Every spirit builds itself a house,
and beyond its house, a world,
and beyond its world, a heaven.
Know them that the world exists for you.
Build, therefore, your own world.

—Frank Lloyd Wright

1.1 INTRODUCTION

People interact with multimedia every day: reading books, watching television, listening to music. We organize and structure this multimedia, such that we can easily access it again. We create photo albums of our holidays, we keep racks of compact discs and tapes with the music we like, we store past editions of magazines in boxes, and use a video recorder to record television programs about topics of our interest. Typically, these multimedia collections end up in old shoeboxes on the attic, guaranteeing pleasure and fun when 're-discovered' many years later.

Since the introduction of multimedia in personal computers, it has become more common every day to digitize part of the multimedia data around us. A major advantage of digitized data over shoeboxes is that digitized data can be shared easily with others. People now create their own homepages on the world wide web (WWW), partially as a tool to manage the information they collect. But, browsing the web makes clear that a computer with a web server is not the best tool to share your 'shoebox data'. It is not

easy for others to find your data, and, the information pointed at by search engines is often incorrect, or has been moved to another location.

A better solution to create large collections of digitized data is to organize the data in (multimedia) **digital libraries**. A digital library supports effective interaction among knowledge producers, librarians, and information and knowledge seekers [AY96]. Adam and Yesha et al. characterize a digital library as a collection of distributed autonomous sites that work together to give the consumer the appearance of a single cohesive collection. A digital library should be accessible through the WWW as well, but it can provide much better support for searching and sharing the data, because it is not completely unstructured like the WWW. The popularity of so-called 'portal sites', and the increasing amount of domain-specific search engines appearing on the web, also indicate that better organization of data available in the WWW is necessary to make it accessible.

This dissertation investigates the potential role of database management systems in software architectures for the creation and operation of multimedia digital libraries. Database technology has provided means to store and retrieve high volumes of data in the business domain. But, database systems have always been designed for the management of alphanumeric data such as names and numbers. Recently, researchers have started to think about 'multimedia databases'. Unfortunately, anything that simply *stores* multimedia data is called a multimedia database. The capabilities of such databases suffice for typical applications of real estate and travel businesses, as these systems only deal with the presentation of otherwise statically used information. But, a *general-purpose* multimedia database management system should provide much more functionality than just storage and presentation. This thesis is an attempt to define what properties can be expected from a multimedia database system.

1.2 SCENARIOS OF USER TASKS

To establish an informal notion of the potential role for multimedia digital libraries in our daily lifes, this section sketches two possible task scenarios. The purpose of these (mainly fictive) scenarios is to outline the complexity of the tasks for which end-users may consult a multimedia digital library.[1]

1.2.1 Journalism

In the first scenario, which is loosely based on the field study performed by Markkula (see [MS98]), imagine a journalist writing an article about the effects of alcohol on driving. Before she can start to do the actual work of writing the article, she has to collect news paper articles about recent accidents, scientific reports giving statistics and explanations, television commercials broadcasted for the government, and interviews with policemen and medical experts.

After the article has been written, she has to illustrate it with one or two photos. She searches in her publisher's photo archives, and probably tries the archives of some stock footage companies as well. Typically, she first generates several illustration ideas. Based on these ideas, she searches and browses archives and catalogs, and prints some of the photos she likes (or writes down their locations). After these steps, she selects

a small set of candidate photos, and eventually chooses the 'best' photos from this candidate set for publication. The selection of 'good' photos from the candidate set is very subjective, and depends mainly on visual attributes of the photos that are hard to describe in words: Markkula reports that journalists used expressions relating to the atmosphere or the feelings perceived, such as 'dramatic', 'surprising', 'affective', 'shocking', 'funny', 'expressive', 'human', and 'threat'. She also reports that often 'non-typical' photos were preferred.

Searching for photos related to proper names or news events is relatively easy. But, finding photos for other illustration ideas can be difficult, such as those showing object types, concerning themes, or photos about places instead of photos taken at those places. During the process of finding a good photo, the journalist prefers to browse through many photos (browsing hundreds of photos is not extreme). Browsing is an important strategy for two main reasons. First, it might lead to new illustration ideas, even if the photos seen are not very relevant for her project. Also, the criteria that define a 'good' photo are difficult to express by words, but easily applied when a photo is seen.

1.2.2 Fashion design

The second scenario focuses on a fashion designer developing a concept for a dress to be worn by receptionists of some big retail office.[2] To succeed in this creative design task, he first collects many different multimedia objects. The designer needs descriptions and pictures of the retailer's products, video fragments of buyers at the premises, photographs revealing details of the entrance and reception area, advertisements in magazines, commercials on television, video and audio fragments of 'vision development breakfasts', and many other pieces of information associated with the retailer. The designer also browses through previous designs, studies preferred dresses from colleagues, and views some videos of recent developments in fashion design.

The user task of this scenario involves the use of large amounts of multimedia data. Fashion designers working alone may not need advanced information technology. Piles on their own desks and shoeboxes filled with old designs may provide easier ways to handle the data than a digital library. But, design tasks are typically performed by teams of designers. Even if these people work at the same time in the same room, they would still need a tool to find what they need in the 'organized mess' of the other team members.

1.3 EXAMPLES OF DIGITAL LIBRARIES

Various digital libraries are being developed in many locations, well-known examples including the Informedia project at CMU (discussed in the next subsection), the U.C. Berkeley project for environmental data of California State, and the University of Michigan library for earth and space materials. This section describes two digital library projects in detail, to describe the type of functionality provided and the type of technology used in existing prototype systems. It is meant to illustrate that the current prototype systems provide already some basic functionality to browse large collections

of multimedia data, even though they cannot support the users of the previous scenarios with all aspects of their tasks.

1.3.1 A digital library for news bulletins

The Informedia project at Carnegie Mellon University [HS95] has developed prototype software for a digital library that will contain over a thousand hours of digital video, audio, images, and text materials. The project has focused on technology that adds search capabilities to this large collection of video data. As shown in Figure 1.1, the Informedia software supports two modes of operation: library creation and library exploration.

The Informedia approach to library creation is to apply a combination of speech recognition and image analysis technology to transcribe and segment the video data. The project uses the Sphinx-II speech recognition system [HRT+94] to transcribe the audio track. The transcribed data is then indexed to accomplish content-based retrieval. Initially, a highly accurate, speaker-independent speech recognizer transcribes the video soundtracks. This transcription is then stored in a full-text information retrieval system.

Speech recognition is not an error-free process and formulating a query that captures the user's information need is very hard. So, not all retrieved videos will satisfy the user's information need. When we want to get a quick impression of a text document, we check the table of contents, take a look at the index, and skim the text to find the pieces of information that we need. But, the time to scan a video cannot be dramatically shorter than the real time of the video. So, some different approach to 'video skimming' has to be supported in the interface. Using image analysis techniques like automatic shot detection in combination with analysis of the speech recognizer's output, the essence of the video content can be expressed in a small number of frames. This small sequence of frames is called a 'film strip'. Using the film strip, fast browsing of the video material is possible.

In the News-on-Demand prototype system [HWC95], a library with television news is created fully automatically using the Informedia technology. Of course, an automatic data collection system based on speech recognition is error-prone. Errors found in experiments with the system include the wrong identification of the beginning and the end of news stories and false words in the transcripts. Despite of the recognition errors, the prototype system shows big changes in the way people will 'watch television' in the future. The system allows us to navigate the information space of news stories interactively based on our interest. Compare this with waiting passively for the next news broadcast, following a path through this space that has been planned by somebody else, beyond our control.

1.3.2 A digital library for cultural heritage

The CAMA digital library[4] is a pioneering project in African culture, coordinated by University of Cape Town. CAMA seeks to create a living network of Arts, artists, and musicians, to preserve the cultural heritage of the continent of Africa. The collection includes both traditional and contemporary artworks of various media types. Parts of

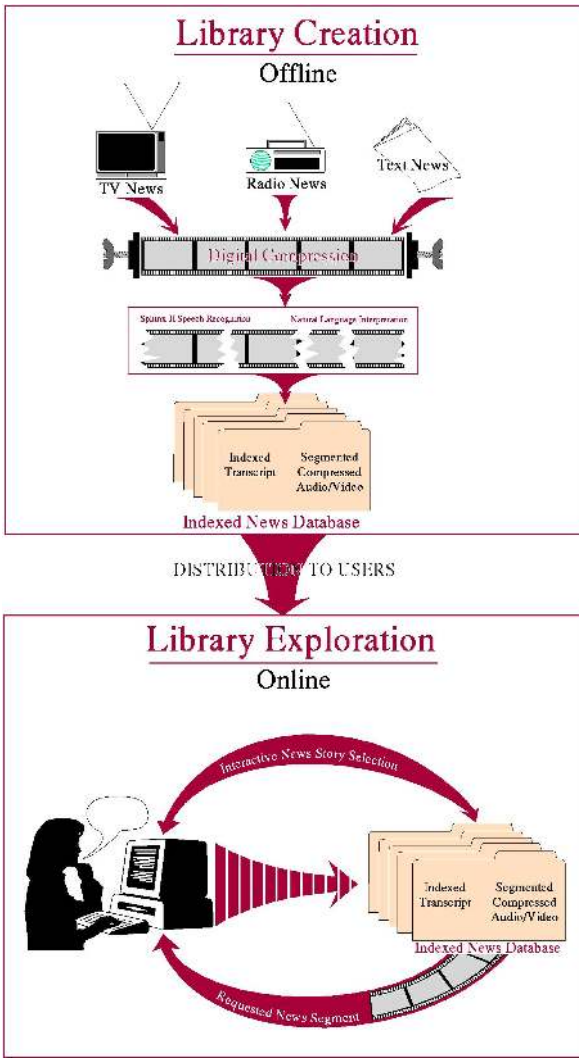


Figure 1.1. The Informedia architecture.[3]

the collection cover over 400 digitized photos of artworks from the Royal Academy of Art's 1995 London exhibition, a collection of stone sculptures from Zimbabwe, a collection of flags from the Fante people of Ghana (from a book by Peter Adler), as well as recordings of traditional folk songs and modern African jazz, produced by Brian Eno at the 'African Alchemy project' during some workshops in Capetown and Johannesburg.

In contrast to the Informedia project, CAMA has concentrated on collecting and archiving multimedia data for African culture, as well as art-historic descriptions of these digitized representations of the Arts and their creators, rather than the devel-

opment of new technology. The main goal of CAMA is to bring images of Africa's artistic heritage 'home' to Africa, albeit in a digitized form. The project is mentioned here to emphasize the potential value of digital libraries for society, as well as its value as a tool to facilitate education and research in the social sciences.

CAMA will keep growing as more and more art is digitized all over the continent, and it provides an excellent basis for historic research. But, the existing technological infrastructure facilitates such research only through browsing web pages, that index the material by category, textual description, and location of origin. Using this collection effectively for scientific and educational purposes will require a more advanced software infrastructure that provides better facilities to access the data.

1.4 WHAT IS THIS THESIS ABOUT?

Building large digital libraries is a problem that challenges most disciplines in computer science. In their overview of 'strategic directions' for digital libraries, Adam and Yesha et al. identify numerous issues that require further research, touching fundamental research questions as well as more practical software engineering problems [AY96]. A huge volume of papers is relevant to at least some aspect of building digital libraries, and these papers are spread over many different fields: operating systems, databases, information retrieval, artificial intelligence (both computational vision, and reasoning under uncertainty), pattern recognition, cognitive science, etcetera.

Most research takes place in a single discipline; but, a software architecture for digital libraries must address many problems, and hence research into building such systems should seek beyond the traditional boundaries of disciplines. Looking back on the scientific literature that has appeared in the last decade, the researchers in different disciplines seem to have reached some local optima, while there is a clear need for integration of the different types of technology developed in these fields. For, there are some obstinate problems with the current state of the art:

- The gap between the functionality required for the user scenarios of Section 1.2 and the user interfaces of the prototype systems is quite big;
- Developing advanced multimedia retrieval applications on top of existing systems is a complicated process;
- The current approach to integration of different components cannot be expected to scale up to data collections of realistic sizes.

This thesis concentrates on the task of *data management* in digital libraries. The underlying hypothesis is that, to enable progress beyond these local optima in different disciplines, better tools are needed to manage collections of multimedia data and control the processes that operate on that data. The objective of this thesis is to investigate how the knowledge about database systems developed for business domains extends to the emerging domain of multimedia digital libraries. This objective is refined in the following research questions:

- Can we identify requirements with respect to data management that are specific for applications in a multimedia digital library?

- If so, can we support these requirements in a subclass of DBMSs (that will be called *multimedia DBMSs*); that is, without violating the design principles (especially the notion of data independence) that characterize 'the database approach' to data management?

- If so, can we provide this support in an efficient and scalable manner?

The research method for studying these questions is to build and analyze a prototype for data management in an example digital library consisting of images.

1.5 SOME COMMENTS ON THE RESEARCH METHOD

The research goals of this dissertation are questions of the type studied in the scientific field of **information sciences**. In the first issue of *Information Systems*, appearing in 1975, Senko defined information sciences as follows [Sen75]:

> *In our discipline, we are concerned with, (1) the efficient use of human resources in the design, implementation, and utilization of information systems, and (2) the efficient utilization of the physical-mechanical resources of the systems themselves. Our goal, therefore, is to search for the* ***fundamental knowledge*** *which will allow us to postulate and utilize the most efficient combination of these two types of resources.*

This research does not attempt to find the single best solution in some particular aspect of database support for digital libraries. Instead, it attempts to create order in the chaos and confusion about what is a 'multimedia database', define a blueprint of such a system, *and* provide guidelines for the implementation of such systems. Of course, this ambition is somewhat problematic from a methodological viewpoint: this thesis not only claims to describe a whole class of systems, these systems do not even exist yet. How can you evaluate the merits of a complete class of database systems, for a problem as ill-defined as multimedia retrieval, without having several example systems to study?

This dissertation alleviates this problem by carefully developing a line of reasoning that incrementally identifies a set of problems with multimedia data management, addresses some of these problems, and returns to the identification of remaining problems. Each step generalizes the solutions taken in current systems, and compares these against currently known approaches. The solutions are unified with the principles of database system design. The result is a framework with which it is possible to build and analyze multimedia DBMSs. By clearly identifying each step, the design decisions are made explicit. The line of argumentation is reinforced by developing a prototype implementation, that demonstrates how the guidelines may be applied in a real system. Still, this prototype is just a single implementation of the class of systems described in the thesis. Also, being a prototype, it does not guarantee that the architecture does not break under different applications than the ones tested, nor wether all its promises can be fulfilled in a real implementation without discovering new problems. As such, the main contribution of this dissertation can only be a *thesis* rather than a proven solution. It is the thesis that the way of thinking put forward in this manuscript provides a guideline for the development of multimedia database systems that are sufficiently powerful that they can support multimedia libraries effectively and efficiently.

1.6 OUTLINE OF THESIS

The remainder of this thesis is organized as follows. Its objectives have been approached in a bottom-up manner: starting at the core of database management systems, the dissertation works its way up to the design of an open distributed architecture for multimedia digital libraries.

Chapter 2 presents the principles of database systems, concentrating on data abstraction, data independence, and efficient query processing. The main purpose of the chapter is to reveal the weaknesses in various popular approaches to extend the scope of traditional DBMSs for data management to other domains than just business applications. It proposes the multi-model DBMS architecture as a promising alternative, and introduces the So-Simple DBMS, a prototype implementation of this new database architecture.

Chapter 3 investigates the problems with the management of multimedia data, that are not addressed well in current database management systems. It discusses different approaches to content abstraction, using various types of metadata. It then introduces the query formulation problem, and formulates four requirements that should be addressed in any multimedia DBMS. As a part of these requirements, it defines the new notion of content independence, a dual of data independence for the management of the metadata used in querying by content.

Chapter 4 proposes the Mirror architecture, an architecture for multimedia DBMSs that addresses these new requirements. It explains the strong relationship between multimedia DBMSs and information retrieval, and generalizes probabilistic IR theory to handle some differences between text retrieval and multimedia IR.

Chapter 5 presents the Mirror DBMS, a prototype DBMS based on the multi-model DBMS architecture, that unifies information retrieval with the database approach by proposing an algebraic approach to IR query processing. It explains the operators that support the implementation of the retrieval engine component in the Mirror architecture, and discusses a prototype image retrieval system, as well as the use of the Mirror DBMS for the evaluation of IR theories on the TREC collection, a large test collection to evaluate the effectiveness of text retrieval. It also discusses some opportunities for query optimization.

Chapter 6 identifies some additional constraints for the implementation of multimedia digital libraries, challenging the traditionally monolithic architecture of database systems. It shows that multimedia digital libraries require an open and distributed architecture instead, and proposes a new type of distributed DBMS in which middleware for interoperability between distributed components is an integrated part of its architecture.

Chapter 7 discusses the evaluation problem of multimedia retrieval by content. It reviews the evaluation performed in many different projects, and identifies common mistakes when the quantitative IR evaluation methodology is used without fully understanding its underlying assumptions. It emphasizes the importance of evaluation in the further development of multimedia digital libraries.

Finally, Chapter 8 summarizes the contributions made with this thesis, and discusses directions for further research.

Notes

1. In the remainder of this thesis, ‘user’ refers to end-user unless stated otherwise.

2. This scenario is not based on a published field study like the previous scenario. Instead, it resulted from some informal, personal communication with Gerrit van der Veer, who had interviewed fashion designers about their work in the past.

3. Figure received from Alexander Hauptmann, and was previously used in [KdVB97].

4. CAMA stands for Contemporary African Music & Arts Archive.

2
ARCHITECTURE OF DATABASE MANAGEMENT SYSTEMS

No change, I can't change, I can't change, I can't change,
But I am here in my mould, I am here in my mould,
And I'm a million different people from one day to the next,
I can't change my mould, no, no, no, no, no

(Have you ever been down?)

—Richard Ashcroft, excerpt from *Bitter sweet symphony*

2.1 INTRODUCTION

Most people have some understanding, although usually rather vague, of what makes a system a 'database system'. This chapter presents more precisely the main characteristics that define a software system as a database system. It is a selective view on the history of databases, zooming in on the issues that are most relevant for this thesis. The ideas discussed are not new; rather, they have been widely discussed in the early seventies, and the success of relational database management systems in the business domain can be attributed to them. However, it often seems as if the essential ideas have been 'forgotten' in the hurry to develop database technology for emerging application domains.

This chapter begins with the characteristics of the database approach, focusing on data independence and the ANSI/SPARC architecture. It discusses the benefits of data abstraction, introduces the relational data model, and explains the role of set-at-a-time

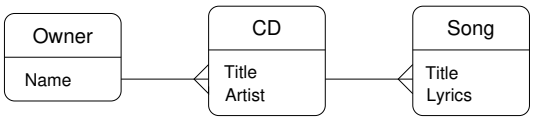


Figure 2.1. The UoD of a compact disc database.

algebraic query languages in query processing. After detailing why current object-oriented and object-relational database systems are likely to have problems with query processing on large volumes of data, the chapter concludes with a presentation of a new architecture for the design and implementation of database systems. The So-Simple DBMS is introduced as a prototype implementation based on this idea; this prototype database management system is used throughout the remainder of the thesis.

2.2 DATA MAKES THE WORLD GO ROUND

Elmasri and Navathe define a database as a collection of related data [EN94]. A database models some aspects of the real world, referred to as the Universe of Discourse (UoD). Assume we want to administrate a collection of compact discs, e.g. to assist with locating a recording when we want to listen to some particular song or artist. In that case, the universe of discourse would consist of compact discs, their owners, album titles, performing artists, owner names, song titles, and maybe even complete lyrics. Sales statistics, although definitely related to a compact disc, are not interesting for the application, and therefore not part of the UoD. A graphical representation of this example UoD is given in Figure 2.1.

A **database management system** (**DBMS**) is a *general-purpose* software system, that facilitates the processes of defining, constructing, and manipulating databases for various applications.[1] The database and the management software together form a database system; database system is also used frequently as shorthand for database management system. The ANSI/X3/SPARC Study Group on database systems stated that the main objective of a DBMS is to *treat data as a manageable corporate resource* [TK78]. A DBMS helps to increase data utilization and to integrate smoothly the data access and processing function with the rest of the organization. It should also enhance data security, and provide data integrity. But, most of all, a DBMS should reduce the amount of work required to adapt software systems again and again in a changing environment.

A DBMS provides this ability to evolve by emphasizing **data independence**: programs that access data maintained in the DBMS are written independently of any specific files. A database management system that provides data independence ensures that applications can continue to run - perhaps at reduced performance - if the stored data is reorganized to accord other applications higher performance.[2] Handling data using a DBMS provides an alternative for traditional file processing. In the file processing approach, each user defines and implements the files needed for a specific application. So, any changes to the structure of a file may require changing all programs that access this file. Different users replicate data in different files, easily resulting in

inconsistencies later on. Conversely, in the database approach, a single repository of data is maintained that is defined once and then accessed by various users.

The following three characteristics distinguish the database approach from file processing (see also [EN94]):

- data abstraction;
- a database is self-contained;
- program-data independence, and program-operation independence;

Data abstraction

Papadimitriou called abstraction 'the essence and raison d'être of databases' [Pap95]. A DBMS raises the level of abstraction for data manipulation above the level of interaction with the file system. It provides users (which can be application programs) with a conceptual representation of the data, referred to as the database schema. This database schema is specified in its **data model**. The data model is the set of concepts that can be used to describe the structure of a database. It specifies logical type constructors such as tuple, relation, set, etcetera. Furthermore, a data model specifies the operations that are permitted on instances of such types. Well-known data models include the relational data model (see Section 2.3), the NF^2 data model (see Section 2.8.2), and object-oriented data models.

A database is self-contained

A database system contains not only the database itself, but also a complete definition or description of the database. This makes databases self-contained, which is necessary to obtain data independence. The metadata that describes the structure of each file and the type and storage format of each data item is stored in the system catalog or data dictionary.

Program-data independence, program-operation independence

The conceptual representation of a database in the data model abstracts from many of the storage and implementation details. As a result, programs do not have to be rewritten when the structure of the files actually storing the data changes, or the code actually implementing the operations evolves: a DBMS provides program-data independence (the data representation may change) and program-operation independence (implementation of operators may change).

2.3 THE RELATIONAL DATA MODEL

The development of relational database systems is *the* major success of the database field. A relational database management system (RDBMS) is a DBMS based on the relational data model. Codd defined the **relational data model** 'to protect users of large data banks from having to know how the data is organized in the machine' [Cod70]. This section reviews its most important features, using Codd's original paper published in 1970; refer to any textbook on databases for more details, e.g. [Dat85] or

[EN94]. Although initially the idea of relational database management systems was perceived too theoretical by the majority of practisioners, prototype relational systems appeared in the late 1970s, and proved that an implementation could be reasonably efficient. The System R [ABC$^+$76] and Ingres [SWKH76] prototypes were the basis of several commercial database products.

The first (non-relational) database systems did not provide much data independence. In many situations, applications could be logically impaired if the internal data representation changed. As a solution, Codd proposed something completely different to all prevailing approaches: present a mathematical model of the data to all users, based on the theory of relations. The relational data model addresses especially the following three types of data dependencies:

- *Ordering dependence*: Applications that take advantage of the stored ordering of a file are likely to fail to operate correctly if that ordering is replaced with a different one.

- *Indexing dependence*: Indexing structures should only affect the execution performance of data access. However, in early DBMSs, application programs had to refer explicitly to indexing structures; these applications must be adapted every time indices come and go.

- *Access path dependence*: In early database systems, data was represented in hierarchical or network data structures. Access to the data used a low-level navigational language on these structures, exposing detailed knowledge of the physical implementation. Application programs would stop working after the representation changed, because they referenced nonexistent files.

Codd's formal model abstracts from ordering, indexing, and access paths. Since data is only accessed through this model, changing these aspects cannot affect the correctness of applications any longer. Note that such changes can of course affect the performance of applications.

2.3.1 Formal definition

The relational model has a rigorous foundation in mathematics. Its formal definition is as follows. Given sets $S_1, S_2, \cdots, S_n$ (not necessarily distinct), R is a **relation** on these n sets if it is a set of n-tuples, each of which has its first element from S_1, its second element from S_2, and so on: R is a subset of the cartesian product $S_1 \times S_2 \times \cdots \times S_n$. R is said to have **degree** n, and S_j is called the j^{th} **domain** of R. A relation of degree one is unary, degree two binary, and degree three ternary.

Date and Darwen summarize the difference between domain and relation as follows: 'domains comprise the things that we can talk about; relations comprise the truths we utter about those things' [DD98]. Domains encapsulate: values of a certain domain can be operated upon solely by means of the operators defined for that domain. Relations, by contrast, expose their internal structure to the user. Exactly this difference makes it possible to perform operations such as joins, which require knowledge of the structure of the relation.

Date and Darwen also emphasize the important distinction between a relation value (relation) and a relation variable ('relvar'). A value is *an individual constant*, e.g. the character 'a'. For the representation of a value, either on paper or in a computer system, a value can have one or more encodings, like 'a', 'a', or 'a', etc., each denoting one and the same value. A value has no location in time or space, and, obviously, cannot be updated; for, then it would be a different value. A variable is a placeholder for an encoding of a value. It *does* have a location in time and space, and can be updated. The SQL statement `CREATE TABLE R ...;` creates a relation variable (relvar) `R`, that holds an empty relation value. This value represents the current state of the world. After an insert, update, or delete, relvar `R` holds a *different* (encoding of a) relation value. A common cause of misunderstandings is that people often say relation when they really mean relation variable.

2.3.2 Database design with the relational model

As an example of modeling data with the relational data model, consider the compact disc example (Figure 2.1). First, we define the domains: album titles (T), performing artists (A), song titles (S), and owner names (O). Assuming that album titles are unique for each artist, some particular collection of compact discs can be represented as a relation $R(T, A, S, O)$. Recall that a relation is just a single value, one of all possible collections of compact discs that can be constructed in this UoD. In a database system, we declare a relation variable C of (relational) type $R(T, A, S, O)$. When we buy new albums and insert their representations in the database, relvar C is updated and refers to a *different* relation value.

A design based on one relation $R(T, A, S, O)$ is not the only possible relational model of the UoD. Here, the representation of a single compact disc is divided over different tuples: as many as there are songs on the disc. An alternative design introduces a compact disc identifier I, and uses two relvars, of types $R(I, T, A, O)$ and $R(I, S)$. This design has less redundancy, but a main disadvantage is the need for a rather artificial identifier I. Other options represent ownership explicitly, in a relvar of type $R(T, A, O)$, or type $R(I, O)$ using the compact disc identifier. Yet another design alternative is to represent the songs on one album as a relation-valued attribute in a relation $R(T, A, R(S), O)$; Section 2.8.2 discusses the consequences of this option.

It is important to realize that neither of these alternatives determines how the data is physically stored; it is very well possible that the DBMS maps each design to the very same internal representation. Although the second alternative had less redundancy, this is redundancy at the conceptual level, which is not necessarily reflected at the physical level.

2.4 THE THREE-SCHEMA ARCHITECTURE

The ANSI/X3/SPARC Study Group on database systems proposed the **three-schema architecture** as a framework for the design of DBMSs [TK78]. This architecture of database management systems, shown in Figure 2.2, is also known as the **ANSI/-SPARC architecture**. The Study Group took the view that *interfaces* are the only aspect of a database system that can be standardized. Its goal is to separate the

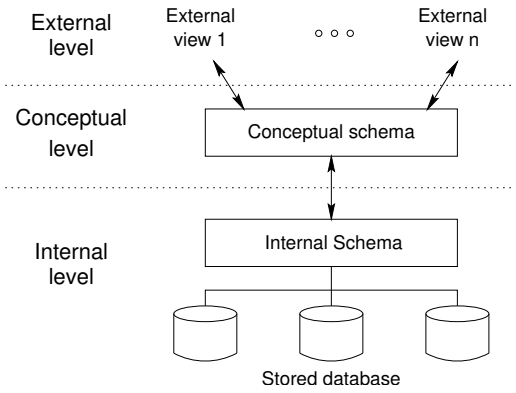


Figure 2.2. The three-schema or ANSI/SPARC architecture.

user applications and the physical database by emphasizing data independence, which insulates a user from the adverse effects of the evolution of the database environment.

The ANSI/SPARC architecture has been developed for database systems that operate in the business domain. Tsichritzis and Klug refer explicitly to concepts like 'the enterprise' and 'line organizations'. Although the ideas related to data independence may very well extend to emerging domains like digital libraries, it remains to be seen whether DBMSs that operate in such emerging domains can and should be designed and implemented according to this architecture. In this section, it is silently assumed that the domain of a DBMS is indeed the business domain. The following chapters will address the suitability of this architecture in digital libraries.

The three-schema architecture recognizes the following three levels in a database system:

- The **internal level** has an **internal schema**, which describes the physical storage structure of the database. It is oriented towards the most efficient use of the computing facility.
- The **conceptual level** has a **conceptual schema**, which describes the structure of the database for its user community, but hides the storage details. The conceptual schema describes a model of the UoD, maintained for all applications of the enterprise.
- The **external level** includes a number of **external schemas** or **user views**. The external schemas are simplified models of the UoD, as seen by one or more applications.

The main contribution of the ANSI/X3/SPARC Study Group has been the recognition that there exists a conceptual level. Tsichritzis and Klug write the following about its purposes:

> *[The conceptual level] should provide a description of the information of interest to the enterprise. It should provide a stable platform to which internal and external schemas may be bound. It should permit additional external schemas to be defined or existing*

ones to be modified or augmented, without impact on the internal level. It should allow modifications to the internal schema to be invisible at the external level. It should provide a mechanism of control over the content and use of the database.

The placement of the conceptual schema between an external schema and the internal schema is necessary to provide the level of indirection essential to data independence. The three-schema architecture provides two types of data independence: **logical data independence** and **physical data independence**. Logical data independence is the property that the conceptual schema can be modified to expand or reduce the database, without affecting the external schemas used in the application programs[3]. Physical data independence allows the internal schema to change independently. Data can be stored at a different place, in a different format, e.g. for reasons of efficiency, without affecting the conceptual schema.

Of course, the *real* data is only visible at the internal level; the other levels only provide a different, more abstract, representation of *the same data*. Thus, the DBMS must establish the correspondences between the objects in the different levels. It transforms a request on the external schema into a request against the conceptual schema, and then into a request on the internal schema. In case of a retrieval request, the results of processing the transformed request over the stored database have to be reformatted to match the user's external view.

The transformations of data are specified in **mappings**, that bind the descriptors in one schema to another. Obviously, these transformations consume processing time.[4] Because of this overhead, few DBMSs have implemented the full three-schema architecture. In DBMSs that support user views, external schemas are usually specified in the same data model that describes the conceptual-level information, causing the 'impedance mismatch' to be discussed in Section 2.7.1. Also, some DBMSs include physical-level details in the conceptual schema. As an example, consider the creation of a table containing alphanumeric data in SQL; this requires the specification of exactly how many characters are used to store the alphanumeric data.

A DBMS based on the three-schema architecture maintains several descriptions and mappings between the levels that are not known beforehand and can change over time. Therefore, a DBMS provides a variety of languages for the specification of schemas and the manipulation of data at different levels of the architecture. Most notable are the data definition language (DDL), which is used to specify the database schema, and the data manipulation language (DML), used to manipulate the stored database. Typical manipulations include retrieval, insertion, deletion, and modification of the data. Finally, the data control language (DCL) is used for managing transactions, access rights, and the creation and deletion of access structures.

In a DBMS in which a clear separation exists between the internal and the conceptual level, the DDL is used to specify the conceptual schema only. Another language, the storage definition language (SDL) is used to specify the internal schema. For a true three-level architecture, we also need a third language, the view definition language (VDL) to specify user views and their mappings to the conceptual schema. Often, these languages are not distinct, but integrated in a single database language, that consists of varying constructs for conceptual schema definition, view definition, data

manipulation, and storage definition. A well-known example of such a language is of course the SQL language.

2.5 DATABASE 'GOODIES'

Some other characteristics of database systems (well-known, but ignored in this survey so far) originate from the fact that a database system typically has many different users, who require the data for different tasks. Multiuser DBMS software has to ensure that concurrent transactions operate correctly without interference. Additional properties include the enforcement of integrity constraints, security and authorization, and backup and recovery. In the database approach, implementation of these facilities - sometimes referred to as database 'goodies' - may be done only once, when building the DBMS: a nice example of code reuse. Application developers do not have to worry about actions of other users, and may assume data recovery after system crashes; the DBMS takes care of this. Because the algorithms involved are usually rather complex, this not only reduces the implementation effort of building applications that access the database simultaneously - it significantly reduces potential errors caused by flawed implementations of such algorithms.

A negative effect of always providing these properties in DBMS software is that emphasizing data independence, seems to have shifted to the background. For example, in the introduction of a special issue of the Communications of the ACM on next-generation database systems, Cattell does not mention data independence at all. Instead, he claims that the important features of relational DBMSs are: the ability to deal with large amounts of persistent data efficiently, using transactions for concurrency control, and recovery.

Another drawback is that DBMS software has grown very large and complex, and the overhead caused by this complexity is not always needed by the applications. Silberschatz, Zdonik et al. therefore argue that database systems should 'break out of the box' [SZ96]. They identify a need for data management in contexts that cannot cope with, or do not need, the overhead of a full-blown DBMS. They suggest that we should reuse database system components, but also consider reusing database techniques and experience in new ways.

This thesis chooses to focus mainly on the roots of DBMSs: data abstraction and efficient query processing. As such, it follows the suggestion of [SZ96], to study the transfer of database experience to other domains in isolation. Chapter 6 brings other aspects of data management back into the picture, like security, concurrency, and rule processing.

2.6 EFFICIENT QUERY EVALUATION

Due to the central role of data abstraction in the database approach, data manipulation can only be described at the abstract level of the data model, where it makes no sense to talk about efficiency: database query languages are high-level declarative languages, that can only express *what* data should be affected, not *how* this should be implemented. Thus, the *efficient* evaluation of expressions in a query language is the responsibility

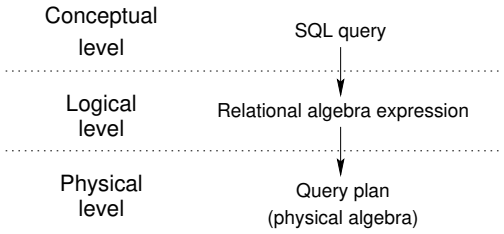


Figure 2.3. Query evaluation in databases.

of the database system. The remainder of this section identifies the techniques applied in the implementation of database systems that enable efficient query evaluation.

Database query languages are usually based on set-theory and applied predicate logic. If there exist only atomic types in the data model, then a first-order predicate calculus suffices. Most end-user languages, including SQL and relational calculus, are based on the following structure, known as the set-comprehension expression:

$$\{f(x) \mid x \in X \ \wedge \ p(x)\} \tag{2.1}$$

Query processing bridges the gap between the database query language and the file system. It transforms requests specified in the database query language into the **query plan**, a sequence of operations in the physical access language. **Query optimization** attempts to determine the optimal query plan - optimal in the sense that the best possible overall retrieval performance is achieved [JK84]. However, in most cases the search space consisting of all query plans that implement the user's original request is too large to be searched exhaustively. As a result, the selected query plan is often only suboptimal. In any implementation of a database system, the task of the query optimizer is more to avoid very inefficient query plans, than to select the one very best option.

2.6.1 Calculus or algebra?

Query languages can be classified using the distinction between a **calculus** and an **algebra**. The difference between the two is that a calculus expression is item-oriented (referring to one item at a time), while an algebra expression is set-oriented [Ste95]. A calculus contains the concept of a variable that represents an item, and thus allows for arbitrary nesting of expressions, whereas nesting cannot occur in set-oriented languages. In set-oriented languages, expressions are context-free and correspond to well-defined execution steps that are mutually independent. In an item-oriented language, variables defined at a higher level can occur free in lower level expressions due to nesting.

The main advantages of an algebraic language are that (1) the join order is not fixed, (2) the intermediate results are clearly visible, and (3) the language is extensible; new operators can be introduced whenever the need arises [Ste95]. A good example of the last advantage is the role of the join in relational algebra. The join operator is not

necessary, for, a join $X \bowtie_{p(x,y)} Y$ is (by definition) equivalent to a selection from the cartesian product: $\sigma_{p(x,y)}(X \times Y)$. The reason to add the join operator is not just a matter of convenience; more importantly, the join can be computed in many different - *more efficient* - ways than the original expression.

To clarify these differences by an example, consider the selection of the titles of compact discs by the artists 'Crowded house', 'Neil Finn', or 'Split Enz', from relation variable C defined in Section 2.3. Let the query set be represented as a relation Q (of type $R(A)$), which contains the artists of interest. In tuple relational calculus, the query is specified as

$$\{\ c.\text{title}\ \mid\ C(c) \wedge ((\exists q)\ (Q(q) \wedge q.\text{artist} = c.\text{artist}))\ \} \tag{2.2}$$

Notice the tuple-variables c and q, that range over relations C and Q, respectively. A naive implementation of this query first ranges over relation C, and then ranges over relation Q for each value c of C.[5] This naive evaluation strategy is better known as **nested-loop evaluation**. A drawback of this strategy is that it is often very expensive, both in time and in space.

An equivalent expression in relational algebra is a sequence of relational algebra operators:

$$\pi_{\text{title}}(C \bowtie_{\text{artist}=\text{artist}} Q) \tag{2.3}$$

From a system's perspective, (algebra) Expression 2.3 is a useful representation. Since the join operator is commutative and associative, it can choose to evaluate either $C \bowtie Q$ or $Q \bowtie C$. The second option can be implemented more efficiently, as an iteration over Q involves only three elements; therefore, a hashtable on $\pi_{\text{artist}}(C)$ can speed up the join's evaluation significantly. This query plan is not so easily derived from (calculus) Expression 2.2, for which the join order has been fixed by the nesting of its tuple variables.

Unfortunately, an algebra is not very suited as a language for users. It is often surprisingly hard to formulate a query in relational algebra, even for quite simple queries. For example, try to find an expression for the selection of pairs of artists that both perform a song, that is not performed by any other artist (assuming that a song is the same if the title is the same). A calculus expression for this request is relatively straightforward to construct; after selecting the pairs of artists that recorded the same song, a negated existential quantifier eliminates the pairs that concern a song also recorded by some other artist:[6]

$$\begin{array}{l}\{\ c.\text{artist},\ c'.\text{artist}\ \mid \\ \quad C(c) \wedge C(c') \wedge c.\text{artist} \neq c'.\text{artist} \wedge c.\text{song} = c'.\text{song} \wedge \\ \quad ((\nexists o)\ (C(o) \wedge o.\text{artist} \neq c.\text{artist} \wedge \\ \quad\ o.\text{artist} \neq c'.\text{artist} \wedge o.\text{song} = c.\text{song}))\ \}\end{array} \tag{2.4}$$

An equivalent algebra expression is not easily found (of course, this is possible for any expression in relational tuple calculus, as relational algebra is relationally complete; i.e., it is at least as powerful as relational calculus). A possible solution is

relation CC computed by Expression 2.5, which requires among others two joins, two selects, and a set difference.

$$\begin{array}{lcl} XY & \leftarrow & \pi_{\mathrm{X,Y.artist}}(\sigma_{\mathrm{X.artist}\neq\mathrm{Y.artist}}(\\ & & \quad (C \text{ as } X) \bowtie_{\mathrm{X.song}=\mathrm{Y.song}} (C \text{ as } Y))) \\ XYZ & \leftarrow & \sigma_{\mathrm{X.artist}\neq\mathrm{Z.artist}\wedge\mathrm{Y.artist}\neq\mathrm{Z.artist}}(\\ & & \quad XY \bowtie_{\mathrm{X.song}=\mathrm{Z.song}} (C \text{ as } Z)) \\ CC & \leftarrow & \pi_{\mathrm{X.artist,Y.artist}}(XY) - \pi_{\mathrm{X.artist,Y.artist}}(XYZ) \end{array} \quad (2.5)$$

2.6.2 From calculus to query plan

Considering this last example, it is not hard to see why end-user languages are preferably item-oriented languages. However, data manipulations at the physical level are preferably performed set-at-a-time, because processing a set of items at once allows the system to optimize in several ways. First, it may perform additional processing (like sorting, indexing, or creation of a hashtable), such that a faster algorithm can be used and the overall performance of the operation is increased. Also, it may avoid duplicate computations for identical items by caching (partial) results. Another optimization is to partition the set and divide the workload accordingly over different processors or different machines.

The set of query processing techniques available in a DBMS form the operators of its **physical algebra**. A query plan is an expression in the physical algebra. This physical algebra is system specific, and has cost functions associated with its operators. For a discussion of a wide variety of set-oriented query evaluation techniques that can occur as operators in a physical algebra, refer to Graefe's survey [Gra93]. The difference between a query expression (consisting of a number of nested blocks, each like Expression 2.1) and an implementation in the set-oriented operators of the physical algebra is quite large. In general, it is easier to derive an efficient query plan from an algebra expression than from a calculus expression. Therefore, it is common to introduce a **logical algebra**, as an intermediate language that can bridge the gap. For instance, a typical RDBMS implementation first translates a SQL expression into relational calculus (both user languages), then transforms the calculus query into a sequence of relational algebra operations (the intermediate language), applies several logical rewrite rules, and, only then, determines the query plan to compute the desired result efficiently [JK84].

A logical algebra is closely related to the data model. It consists of a limited number of operators, that should be relatively easy to map to the physical algebra, but still be sufficiently expressive to describe queries in the data model. The logical and physical operators can differ in a couple of ways. A common difference between the two is that the physical algebra uses multi-set semantics of relational operators. Only when the final results are presented, an extra unique operation is performed (which exists only in the physical algebra). Similarly, a project in a join implementation usually does not perform duplicate removal either. Also, while a logical join operator is symmetric, the operators in the physical algebra that implement joins (such as a nested-loops join,

a merge-join, or a hash-join [Gra93]) are asymmetric (which made it attractive to evaluate $Q \bowtie C$ instead of $C \bowtie Q$ in the example given before).

A common strategy for query optimization is the application of heuristic rewrite rules to expressions in the logical algebra. A well-known example is the 'push-select-down' pattern, in which a select on the result of a join is 'pushed through' the join, such that the intermediate join result is smaller, and therefore evaluates more efficiently: $\sigma_{p(y)}(X \bowtie_{q(x,y)} Y) \Rightarrow X \bowtie_{q(x,y)} (\sigma_{p(y)}(Y))$.

Rewriting logical algebra expressions alone is not a sufficient optimization technique though. Both [Ste95] and [JK84] emphasize the importance of finding a 'good' initial algebra expression *during* the translation from the calculus to the logical algebra, instead of relying on the rewriting of inefficient algebra expressions. A sequence of algebra operations hides many optimization opportunities that are more easily derived from the original calculus expression (especially with respect to equivalent subexpressions, which may be easily 'overlooked' in a long sequence of algebra operators, but can often be detected without problems during the transformation from calculus to algebra). Query processing should be like finding your way to a museum in a strange city using a tourist map: the main streets are shown on the map (the user's query), but sometimes it is smarter to take a shortcut that is not on the map (query optimization). Phrased in terms of this metaphor, rewriting logical algebra expressions is like having a map of such a coarse granularity that you do not dare to leave the main roads, afraid to get lost.

2.6.3 Efficiency, another argument favouring data abstraction!

Data independence has always been the driving force behind the development of database technology. However, apart from data independence, a methodology to obtain efficient query processing is another big advantage of the database approach. As described above, efficient query processing can be achieved using a divide-and-conquer strategy, in which the original information request is transformed into the final query plan using a number of intermediate representations. Each step uncovers another piece of abstraction, and gets closer to the machine level. These transformations enable the application of optimizations based on set-oriented processing, to avoid inefficient nested-loop processing. Of course, not all queries are processed efficiently using a DBMS. But, a database expert can often resolve efficiency problems easily, by either creating an index structure for the 'right' attributes, or helping the optimizer a little bit, e.g. by manually rewriting a nested SQL query into a join query.

A divide-and-conquer strategy using abstract representations of data and queries becomes even more important when the database system supports scalability, using parallel execution and data distribution. The design of parallel versions of algorithms is a complex matter. Parallel query processing has to make decisions about data allocation, data fragmentation, and pipelining between and within operators [Wil93]. Comprehension of 'good' and 'bad' strategies is easier to grasp for a restricted set of algorithms (such as the physical algeba of a database system), than for the general case of any algorithm implemented in a general-purpose programming language. Similarly, data abstraction is necessary to control query processing in distributed database systems. Designing distributed databases makes extensive use of algebraic represen-

tations in the translation from global queries to fragment queries, both for proving correctness of the translation, and for deriving efficient query plans [CP85]. The effect of distributing data over different servers can be studied by taking the communication costs into account during query optimization.

Another potential advantage of putting a lot of implementation effort in a limited set of algebraic operators arises from the complexity of the hardware architecture of modern workstations. Boncz demonstrates (both theoretically and experimentally) how the implementation of an efficient join operator in the physical algebra of the Monet database system should *really* take into account the amount of level-two cache for maximal performance [BMK99]. Directly estimating the effect of the hardware architecture on the implementation of applications in high-level programming languages seems too complicated; expressing the application logic in a sequence of highly optimized physical operators looks like a more viable alternative to get the most out of such hardware developments. Also, whenever changes in the hardware lead to a more efficient implementation of the join algorithm, the application logic will benefit automatically with increased efficiency, without any extra implementation effort!

These advantages with respect to efficiency are a major incentive to investigate the use of query processing techniques in other domains than business applications. Some case studies have already proven that set-orientation and indexing are beneficial in other domains as well; sometimes, these techniques improve the performance of known algorithms well beyond the efficiency of specialized toolboxes, programmed by skilled programmers:

- Seshadri demonstrates a significant improvement in the performance of query processing in sequence databases [SLR96]. His improvement is based on operator pipelining, which was possible because the queries were expressed in a (domain-specific) algebra on sequences.

- Nes et al. report a performance improvement of an order of magnitude using an algebraic formulation of a computational vision algorithm for edge detection [NKJ96]. Here, the speed-up can be attributed to the advantages of set-oriented processing in general, and the use of an R-tree in particular, which makes it possible to benefit maximally from locality of reference when clustering points into edges.

- Goyal et al. argue that even GUI programming can and should be studied as a database problem [GHK+96]. They apply a declarative language in a data centric architecture for GUI programming, and demonstrate a performance improvement through incremental repainting. Since a declarative program allows the compiler to deduce properties such as monotonicity, it is possible to limit repainting to a small subset of the screen.

2.7 OBJECT-ORIENTATION AND DATABASE SYSTEMS

The suitability of a DBMS for some application is closely related to the expressiveness of its data model. The data model of the conceptual level should fit the universe of discourse, since end-users have to understand this model of the real world in order to formulate their queries. A record structure fits best when the population

is homogeneous, i.e., all items have the same fields. Because this is often the case for business data (Stonebraker classifies business applications as 'simple data with queries' [SM99]), relational database management systems have become a standard tool in business database processing. But, not all information of business applications is naturally represented by collections of records. Kent collected many problems concerning record structures in [Ken79], 'as a resource to defend alternative models'. He illustrates how the assumption of homogeneity is often not valid: not all employees of a multinational have social security numbers, and company cars can be assigned to both employees and departments. Also, from an information modelling perspective, it is not clear how 'entity' and record should correspond, or similarly, 'relationship' and record.

New application domains need effective and efficient management of large data volumes as well: scientific data analysis, computer aided design, and - the topic of this thesis - multimedia digital library systems. The data structures encountered in these domains are far more complex than business data, and do not map easily to collections of records. This data is therefore referred to as 'complex data' by the database world. If applications in these domains require ad-hoc query facilities, Stonebraker classifies them as 'complex data with queries'. Languages for data definition and manipulation, such as relational calculus and SQL, have been designed to make it (relatively) easy for the DBMS to process expressions in these languages efficiently. Although the database approach is equally desirable in the emerging application domains, restrictive languages like relational calculus make application development and maintenance unbearingly cumbersome.

2.7.1 The 'impedance mismatch'

In an object-oriented programming language, operations and data are combined (and usually encapsulated) in objects. Object-oriented programming has evolved as the preferred paradigm to develop applications that manipulate complex data. Objects are not flat data structures, but can be nested arbitrarily deeply. Therefore, a collection of objects is in general not easily represented as a table of records. Applications that use a relational DBMS for the management of persistent data, but have been developed using object-oriented programming languages, must 'disassemble' their nested data structures into atomic components, and store these components in the DBMS. Retrieval requires re-assembling the components into the original (nested) data structures. The requirement of these additional steps is often referred to as the **impedance mismatch** between application programming languages and relational database systems [LLOW91]. Notice that the notion of impedance mismatch also refers to another aspect of the interface between programming languages and database systems: the difference between item-oriented thinking encouraged by imperative programming languages, and the set-oriented approach enforced upon the application programmer by database languages.

The impedance mismatch affects not only application development. Ad-hoc querying of objects becomes almost impossible, because users do not know how the components are mapped to the relational schema; this knowledge is encoded in the application logic. Also, a single query at the object level generates a series of queries in the rela-

tional DBMS. Its query optimizer must know the relationship between these requests to determine efficient query execution plans. Performing such processing outside the scope of the database system causes a serious performance degradation, see e.g. [dVEK98].

A DBMS that implements all interfaces of the ANSI/SPARC architecture does not necessarily suffer from the impedance mismatch: the external schema binds the application's data requirements to the conceptual schema. The impedance mismatch between object-oriented application programs and commercial relational database management systems exists because these systems force the external schema to conform to the relational model as well.

Object wrappers provide an object-oriented external view on top of relational databases [CD96]. An object wrapper is *not* a part of the DBMS, but a separate layer of software. It generates classes that act as proxies for data in the underlying database. This reduces the impedance mismatch with respect to programming, but the performance problem remains; unless the wrapper is tied closely to some specific DBMS, encoding specific knowledge about its optimizer and tuning generated queries accordingly. Drawn into extremity, this implies that query optimization is carried over from the DBMS into the object wrapper, which is clearly undesirable from a design viewpoint. Besides, it is unlikely from a commercial viewpoint, since these object wrappers typically proclaim independence of the underlying DBMS.

2.7.2 New generations of DBMSs

As an alternative, database researchers have started looking into database systems based on different data models, and developed query languages for these models. Two broad categories of new generation database systems can be distinguished [Cat91]: those that originated in persistent programming languages, and those that evolved from relational database systems. Systems of the first category, including O_2 and ObjectStore, are generally referred to as object-oriented database systems (OO-DBMSs). Systems of the second category are extensible database systems and object-relational database systems (OR-DBMSs), examples of which include Starburst and Postgres.

Object-oriented DBMSs. The impedance mismatch resulted in a desire to design programming languages in which persistence is orthogonal to type. [Atk89] attempts to describe a set of 'golden rules' that define a database system as object-oriented. In database systems conforming to these rules, the client applications and the database server are tightly integrated. There is no such thing as a database schema; instead, objects in client applications can be declared persistent, and the database system takes care of this persistence. The focus on persistency orthogonal to type is taken one step further in persistent programming languages [AB87]. A persistent programming language adds support for persistency to a single language. An OO-DBMS usually implements several language bindings [CB97], and therefore cannot ever achieve complete orthogonal persistence.

A central notion in object-orientation is the encapsulation of the data structure inside objects. Encapsulation clashes with the notion of data abstraction.[7] As a result, *ad-hoc* query support is difficult to provide, because the user can only use predefined

methods to access the data inside. OQL, a declarative query language developed for O_2, and now part of the ODMG 'standard'[8], therefore breaks with the principle of encapsulation [Clu98]; this option is further discussed in Section 2.8.1.

The main disadvantage of OO-DBMSs is that the data model is part of the application's source code. Indeed, this makes it much simpler to develop a single application requiring persistency of its data. But, sharing data between different applications becomes much more complicated: the design requirements put forward in [Atk89] forgot about the whole issue of data independence. The persistent objects stored in the database are defined for some *specific* application, and this application did most likely not take into account the requirements of other applications that require access to the same data.

Extensible DBMSs. The first implementations of RDBMSs restricted the available data types to a rather limited set including mainly numbers and alphanumeric data. The 'limited number of data types in the relational model' occurs often as an argument in favour of new data models. These data models typically provide extensibility with new data types, along with other features such as inheritance. However, as Date and Darwen point out clearly in [DD98], extensibility with user-defined data types does not require a new data model *per se*. The definition of domain says *nothing* about what can be physically stored. Conversely, since the relational data model does not limit data to numbers and sequences of characters, the implementation of any DBMS that claims to support the full relational data model *must* allow arbitrary data types!

Extensible relational database management systems support **abstract data types** (ADTs). An ADT adds new base types or operations to the database system: it defines new domains, or extends existing domains with new operators. In many extensible database systems, the (expert) user can also add new access structures that support these data types; a common extension is an R-tree for indexing multi-dimensional data [Gut84].

Prototypes of extensible DBMSs included Postgres [SRH90, SK91] and Starburst [HCL+90, LLPS91]. This functionality has now become common in commercial systems, under names varying from 'datablade' to 'data cardridge' to 'user-defined functions' (UDFs). In the following, unless specified otherwise, this thesis assumes that a relational system is indeed extensible with abstract data types.

Object-relational DBMSs. Another development in database technology, also motivated by requirements found in new application domains, is the evolution from relational to object-relational database management systems (OR-DBMSs).[9] These systems are extensible relational DBMSs that support a richer data model than purely relational. Typical features of these data models include references, set-valued attributes, and type inheritance, but it is not clearly defined what makes a system object-relational. Several proposals exist, both in literature and commercial DBMSs, but the suggested functionality differs a lot among them.

Most proposals and/or implementations agree on the support of user-defined types, user-defined functions, some degree of nesting, and type inheritance. Date and Darwen stop right there, and they are already reluctant about inheritance. Their proposal,

published in [DD98], is a plea for basing object-relational database systems on a precisely defined data model, that should be nothing but a minor adaptation to the original relational data model. They argue (quite convincingly) against SQL as a universal database language, and propose a different language D that is strongly related to the proposed data model. However, their 'foundation for true object-relational DBMSs' is not more than a blueprint of the conceptual level (not less either though), and this blueprint ignores completely any issues related to implementation and efficiency.

In [SM99] - on the other end of the spectrum ranging from theory to practice - Stonebraker describes the advanced facilities provided in the commercial relational database products that rule the DBMS market at the end of the 20th century. He emphasizes implementation issues, varying from query optimization to the integration of an OR-DBMS with middleware and applications servers. Apart from the aforementioned properties of object-relational database systems, Stonebraker suggests two other extensions to the data model: references based on object identity, and type casts. Not surprisingly given their devotion to the relational model, Date and Darwen oppose strongly against the support of references based on object identity, which they refer to as 'the second great blunder'.[10] Type casts are also proscriped by their proposal. Finally, Stonebraker discusses rules, but this feature seems rather orthogonal to the choice of data model, and is therefore not further discussed in this thesis.

2.7.3 The conflict between encapsulation and query processing

Despite of controversy and a lack of consensus about what 'object-relational' really means, object-relational systems are the most significant players in today's database world, and they are expected to keep this position [CD96]. It is important to recognize that designing these systems is not understood to the level of relational database systems. Evaluation experiments with the Bucky benchmark, designed to evaluate especially the extra features of the data models in OR-DBMSs, showed that a pure relational schema achieved much better performance in most cases than a schema using object-realtional features such as set-valued attributes [ACD+97]. Also, it is not clear how parallelism can be supported for object-relational queries.

A major problem with the design of (object-oriented) extensions in database systems is that extending the DBMS often results in reduced physical data independence. In most OO-DBMSs, there is no conceptual level that separates the application from the stored data. The OO-DBMS is not aware of the structure of complex objects, but manages black boxes instead. The same argument applies to data manipulated within ADTs in extensible and object-relational database systems, e.g. a datablade that provides image manipulations written in C++. Reduced physical data independence seriously cripples a DBMS in its task to provide performance. The DBMS cannot look insight the objects, causing query evaluation to result in naive nested-loop evaluation. Although encapsulation is a very powerful concept in object-oriented design methods and programming languages, the question arises whether encapsulation should be taken all the way down to the software systems that manage complex data.

Advocates of OO-DBMSs and OR-DBMSs claim 'simplicity' and 'ease of use'. However, it is unclear whether this desired simplification is really achieved without giving up performance. The data mining study performed by Sarawagi et al. is a

good example to support this statement [STA98]. They evaluate several architectures for mining associations with a relational DBMS. One candidate solution is a pure SQL formulation of the mining algorithm, but this turns out to be far too inefficient. Relational databases were not designed with data mining as a target application, but assume an environment with many small updates. Applications in data mining are query intensive instead, causing inefficient query processing on relational DBMSs.

The efficient solutions in their case study are all (at least partly) non-SQL, of which two clearly outperform the others. The best solution first caches all data into the file system, and then uses a special-purpose algorithm on the cached data. Obviously, this cannot really be considered a database solution. Interestingly, the same algorithm implemented completely inside the DBMS, as a UDF, did not perform well; its efficiency was comparable to stored-procedure or SQL cursor-based solutions, but its development involved much more work to implement and debug the algorithm.

Mixing pure SQL with UDFs for part of the algorithm came out as the second-best strategy (and best for some of the reported results). The fastest approach involved one UDF for partitioning the data set in memory and encoding these partitions in binary large objects (BLOBs), and another UDF for computing intersections between these encoded partitions. Another strategy also encoded nested relations in BLOBs, pushed a group-by operation into the UDF. In one variant, the UDF even computes a join with a complete table passed as a BLOB (for each tuple!). Apparently, coding some database operations in the UDFs manages to work around the inefficiency of the pure SQL solution, but simplicity is far to seek. The DBMS already knows how to intersect two sets, how to group a set of tuples, or how to compute a join. The code in the UDF duplicates that code, making it faster for this particular algorithm. Not all these 'improvements' can be parallelized by the DBMS, and operator pipelining between the UDFs is impossible. Also, the integration with the DBMS is too tight: the relative performance of the different options depends heavily on (a particular release of) the DBMS, and is certainly unlikely to generalize to object-relational systems with a different implementation of the internal level.

Summarizing, encapsulation of data and operations inside objects or ADTs affect query evaluation: optimization by the DBMS becomes infeasible, and query processing too often resolves into object-at-a-time evaluation. These problems are more severe in distributed databases, or query processing on parallel machines. The so-called 'next-generation database systems' have moved onto the shoulders of application developers the burdens of finding the optimal query plan and parallelizing code that processes large volumes of data. Small proof-of-concept applications may very well be simpler to build with an OO-DBMS. But, it is unlikely that these applications will scale up to large volumes of data. Apparently, there exists a trade-off between simplicity of data management (strived for in the database approach) and simplicity of application development (strived for in persistent programming languages). So, simply adding ADTs to existing relational technology is not a sufficient solution for the support of non-traditional applications.

2.8 EFFICIENT OBJECT QUERY EVALUATION

Section 2.6 discussed the benefits of data abstraction for efficient query evaluation. The question arises whether a rewrite approach via logical algebra into physical operations, the foundation of query optimization in relational query processing, is also possible for an object query language like OQL. This section addresses that question for a particular subclass of object-oriented data models that does not enforce encapsulation in the specification of data structures. The nested relational data model is introduced, followed by the formulation of NF^2 algebra. It concludes with a discussion of a rewrite procedure and logical algebra targeted to efficient query evaluation.

2.8.1 Structural object-orientation

Dittrich introduced the following classification of object-orientation in database systems (in his terminology, full object-orientation attempts to combine the two approaches) [Dit88, Dit91]:

- *Structural object-orientation*: the data model provides mechanisms for the specification of highly structured entities.
- *Behavioral object-orientation*: arbitrary user-defined, type-specific operations can be associated with data entities.

Structural object-orientation distinguishes between **base types** (or atomic types) and **structured types** (or non-atomic types). Base types correspond to domains in the relational data model. Structured types are made up of components, assembled by applying the available **type constructors** recursively at various levels. Examples of type constructors are set, multiset, list, tuple, and array. Type constructors should be **orthogonal**; in a structural object-oriented data model (or **complex object model**) that supports sets and tuples, a set of sets is a valid type, just as a tuple of sets or a tuple of tuples.

Structural object-orientation implies the definition of complex data structures without encapsulation. Hence, it does not rule out optimization by the DBMS; it can look inside the objects to determine efficient query plans. Query evaluation in a DBMS that supports structural object-orientation does not automatically imply nested-loop evaluation. For example, O_2 has a structural object-oriented data model that can be queried with its query language OQL. An optimization technique that has been applied succesfully on OQL queries is the transformation of a path expression into a join, e.g., compute `c.owner.name` as $\pi_{\mathrm{name}}(C \bowtie_{\mathrm{ownedby}=\mathrm{ownerid}} O)$.

2.8.2 Nesting and the relational data model

The relational model provides only two type constructors: relation and tuple. These type constructors are not orthogonal, because a relation always consists of tuples. Hence, a complex object model that supports set and tuple constructors includes the relational model as a special case, but the reverse is not true.

A wide-spread belief about the relational data model is that all data *must* be in the first normal form (1NF). The original presentation of the relational data model

did however not exclude non-atomic domains; when discussing the possibility of non-atomic domains, Codd gives an example of the domain 'salary history', which consists of all binary relations between domain 'date' and domain 'salary' [Cod70]. About normalization, Codd writes only that 'the possibility of eliminating non-simple domains appears worth investigating'.

Several reasons justify the elimination of non-atomic domains. First, arrays are an efficient implementation of relations with simple domains. Second, the array representation is a convenient way to exchange bulk data between systems with different internal representations. Also, most relational theory (in particular theory that helps to design 'good' relational schemas) has been developed with the first normal form assumption in mind. Finally, if all relations are in first normal form, a high-level declarative query language on these relations can be based on first-order predicate calculus. Relational algebra and relational calculus are only useful if all relations are in first normal form, as there are no operators that can access relation-valued attributes.

2.8.3 Nested relational algebra

The relational data model with the restriction that all relations are in first normal form is also known as the flat relational model. If this restriction is removed, the data model is usually referred to as the **nested relational model**, also known as the NF^2, the XNF, or the N1NF data model. Now recall the example database of compact discs and their owners. For the nested relational design $R(T, A, R(S), O)$, selecting compact discs containing a song called 'De verzoening' is impossible with relational algebra: we need a more powerful language that allows operators to move into relation-valued attributes.

A well-known algebra for the nested relational data model is the NF^2 algebra defined by Schek and Scholl [SS86]. NF^2 algebra augments relational algebra with the nest (ν) and unnest (μ) operators, which convert between nested and flat relations. The nest takes a set of attributes, and forms a new relation-valued attribute out of these; the unnest consumes a relation-valued attribute and replaces it with its set of attributes. The unnest is inverse to the nest operation. But, unnesting cannot generally be inversed by nesting, for empty relations can occur as attributes in nested relations. Therefore, NF^2 algebra cannot be defined by simply unnesting all relations into the first normal form, applying some basic algebraic expressions, and converting the result back to NF^2 relations.

[SS86] shows that NF^2 algebra should minimally extend relational project to allow the application of the other operators within the projection list. With some minor rewrite rules, the resulting algebra can directly apply any algebraic operation that is defined on flat relations to relations within relations as well. As an example, the following NF^2 expression retrieves the compact discs containing a song 'De verzoening' from a relvar C' of type $R(T, A, R(S), O)$:[11]

$$\pi_{\mathrm{title,artist}}(\sigma_{\mathrm{match}\neq\emptyset}(\pi_{\mathrm{title,artist},\sigma_{\mathrm{song}=\text{'De verzoening'}}(\mathrm{songs}):\mathrm{match}}(C'))) \qquad (2.6)$$

The select in the project list constructs a subset (called match) that is only not empty if the requested song occurs on the compact disc. The outer select retrieves the compact discs for which this set is not empty.

2.8.4 Query evaluation and structural object-orientation

Steenhagen's thesis addresses query processing of high-level declarative object query languages like OQL [Ste95]. She states that research concerning NF^2 algebras has not focused on their function to facilitate efficient query processing, but only on the definition of 'some' algebra that can handle set-valued attributes. Most proposals for nested relational algebras, including [SS86] discussed before, intend to use NF^2 algebra at all levels of the database architecture. But, a mapping that leaves the nested expressions as they are does not help to find an efficient query plan at all. For example, the select inside the project in Equation 2.6 is not evaluated set-at-a-time, unless it can somehow be processed for all relation-valued attributes simultaneously. Expression 2.6 does not help to achieve this behaviour.

Steenhagen develops a logical algebra called ADL, specifically targeted as an *intermediate* language that helps to derive efficient query plans. Her work is based on the following underlying assumptions:

- the intermediate language should be an algebra;
- query optimization should play a role in each phase of the implementation process.

She shows that a set of operators in NF^2 algebra is not sufficient to derive efficient query plans from expressions specified in an object-calculus. OQL is an orthogonal language, so expressions can be nested arbitrarily: not just in the where-clause, but also in the select-clause, and operands can be tables as well as set-valued operands. Grouping is often necessary in the translation from OQL to complex object algebra, to avoid the 'complex object bug', a problem analogous to the infamous count-bug of relational systems[12]. Without extra operators, evaluation of NF^2 expressions results in many inefficient nested-loops. As a solution, Steenhagen introduces the **nestjoin** operator, which performs grouping within the join, and claims that this grouping can be implemented efficiently in existing join algorithms. Her translation algorithm introduces set-orientation by replacing nested iterations with joins and nestjoins when appropriate.

She concludes that query processing in a structural object-oriented data model allows *in principle* the same strategy for query evaluation as has been applied successfully for years in relational databases. By adding non-standard join operators to NF^2 algebra, the logical algebra can be evaluated with efficient physical algebra operators. Steenhagen has not worked on an implementation of her approach, and therefore cannot provide a cost model; but when her top-down approach is augmented with a bottom-up approach based on a cost model, rewriting OQL into efficient query plans seems feasible.

2.8.5 Query evaluation and behavioral object-orientation

Structural object-orientation is not always a proper solution. Some operations (usually *domain-specific* operations) can only be implemented efficiently in a general-purpose programming language. To support applications that perform such operations on large volumes of data, a DBMS must support behavioral object-orientation. Of course, extensible and object-relational databases support this functionality using ADTs. But, ADTs in extensible and object-relational DBMSs are 'blackbox' ADTs of which the DBMS only knows the signature. This inhibits the query optimizer to use the semantics of operations defined in the ADT.

Consider an expression `Clip(Sharpen(Image), Region)`: the image is sharpened, and subsequently a subarea is selected. Sharpening the image *after* clipping has the same effect, and is probably more efficient: especially if the image is much larger than the region to be selected. In this particular example, the reader may think of a user that poses the query in this inefficient manner as rather 'silly'. But, such an inefficient expression may well have been generated by a graphical user-interface. Whatever the source of the order of method invocations, the DBMS should figure out to first clip and then sharpen.

Seshadri proposes a database architecture that addresses this problem, based on the idea of **enhanced ADTs** (**E-ADTs**) [Ses98a]. E-ADTs expose some part of their semantics to the DBMS, such that the DBMS can optimize expressions involving a combination of method invocations, like the previous example. In the prototype implementation of the *Predator* DBMS, an E-ADT can implement an optimization interface to optimize the query plan using its own algebra, it can perform the evaluation of a query plan, it can extend the catalog with its own schema information and statistics, and it can provide multiple physical implementations of values of its type.

Supporting E-ADTs requires quite a different architecture of the DBMS. *Predator* is a multi-threaded query processing engine built on top of the Shore storage manager [Ses98b]. Instead of a complete database system, it is an architectural *framework* in which E-ADTs are plugged in; the resulting system can be viewed as a collection of query processing engines. The framework defines the interfaces of tasks involved in query evaluation, but the E-ADTs provide the implementations. For example, an E-ADT can implement the `Optimize()` interface, which then becomes part of the optimization process. *Predator* also provides a rule engine such that E-ADTs can declare optimizations that are easily expressed as heuristic rules (like 'perform clip first').

An interesting aspect of *Predator*'s design is that E-ADTs are modular extensions, like normal ADTs. A query expression consists of several blocks, each of which is delegated to the responsible E-ADT by a central coordinator. The only requirement is that the boundaries between subexpressions for different E-ADTs can be detected by the system. In the example given before, the image-related part of the parse tree is delegated to the image extension; the image E-ADT knows that swapping the invocations of clip and sharpen is a good idea, and rewrites this part of the query. The relational model is supported in 'just' another E-ADT, that allows *Predator* to support nesting of relations as well. Also, domain-specific query languages can be supported, e.g. for sequences, and mixed with language constructs from other E-ADTs [SLR96].

The relational E-ADT makes *Predator* a 'real' DBMS. Its functionality is sufficient to run the Wisconsin benchmark and the TPC-D benchmark. But, *Predator* is not primarily assumed to be a relational database engine. The basic framework with an image E-ADT may be very useful as a stand-alone image manipulation toolkit.

The *Predator* architecture does not address all problems with encapsulation identified in Section 2.7.3. Parallelization is possible within an E-ADT, and materialization of intermediate results can be avoided within an E-ADT. But, it is not clear how these optimizations can cross the boundaries of several E-ADTs. For example, operator pipelining spanning different E-ADTs is not supported. Also, storage management has to be implemented directly on the Shore storage management library, increasing the risk of duplicated functionality inside implementations of various E-ADTs.

The orthogonality between the relational E-ADT and the other E-ADTs may cause efficiency problems in schemas with several levels of nesting. For, data from inner levels is delegated to different E-ADTs, and may be scattered all over the disk, complicating set-oriented processing. This can be illustrated with an example concerning the representation of video data as a sequence of scenes. Assume that each scene is modeled as a structure containing a relation of the actors starring in the scene, and a sequence of shots, each of which is represented by some keyframes. A possible query requests thumbnail images of the keyframes stored for scenes starring 'Nicole Kidman'. Such a query concerns both the relational E-ADT (for the selection of appropriate scenes), and the image E-ADT (for retrieving and resizing the keyframes). Since neither E-ADT is 'in charge', set-oriented processing of the requested keyframes may be hard to obtain, resulting in inefficient nested-loop query evaluation.

2.9 THE MULTI-MODEL DBMS ARCHITECTURE

Summarizing Sections 2.7 and 2.8, it is clear that non-business applications are not well supported by current database architectures. The design of OO-DBMSs makes query optimization infeasible, and complicates sharing of data between applications because it does not emphasize data independence. The ADT mechanism in extensible relational database systems conflicts with optimization in a similar manner. Using ADTs in an efficient manner forces programmers to perform database operations such as joins within the ADT, which causes new efficiency problems when e.g. such a join should have been parallelized. In the extreme, programmers implementing ADTs will build a complete DBMS kernel inside each ADT. The E-ADT paradigm is an answer some of the problems with ADTs, but, as there is no global overview accross E-ADTs, inter-operator optimization is impossible, and parallelism and distributed query evaluation have to be handled in each E-ADT separately.

2.9.1 The open implementation principle

Kiczalis has identified that many performance problems in different types of software systems are caused by *blackbox abstraction*, the basic design principle of software engineering [Kic96]. It is simply impossible to hide all implementation issues behind an interface, because the issues related to the implementation-strategy will show through in performance. This often leads to either 'coding between the lines', or code duplica-

tion, which Kiczalis calls the hematoma of duplication. Kiczalis proposes a different methodology, that he calls the **open implementation** principle. Open implementation still uses abstraction, but lets clients assist with or participate in the selection of the implementation strategy. A good example is operating system support for virtual memory management; modern operating systems provide the client with the possibility to advice on the buffer-policy.

Relational database systems can be viewed as systems that have applied the open implementation principle for many years. The client, which decides about the implementation strategy, is the query processor. This component chooses, based on its cost model, the physical algebra operator that implements the logical operator most efficiently. This relationship between the database query processing and the open implementation principle may explain the efficiency problems caused by blackbox ADTs discussed in Section 2.7.3. For, the operations specified in the ADT do *not* allow the query processor to participate in the decision for the implementation strategy. By violating the open implementation principle (that clients should participate in decisions about the implementation strategy), efficiency problems can be expected. E-ADTs have been proposed as a solution, but problems remain with requests that cross the boundaries of several extensions, because these boundaries between E-ADTs are blackbox abstractions.

2.9.2 Different data models at different levels

This section proposes the **multi-model DBMS** architecture as an alternative design for database systems in non-traditional application domains. The design of a multi-model DBMS has a layered architecture, with a central role for data abstraction. The unique, distinguishing aspect of this architecture is that the conceptual data model used by its end-users is mapped to a physical implementation *using different data models at different levels of the database architecture*. Although this idea may be a good basis for the design of extensible relational systems as well, the choice made for the So-Simple DBMS is to provide a structural object-oriented data model at the logical level, and map this to a binary relational model at the physical level. The first prototype of a multi-model DBMS has been developed by Annita Wilschut, in the MAGNUM project. The prototype, referred to in this thesis as the So-Simple DBMS, is discussed in Section 2.10. This section motivates the ideas at the foundation of this architecture, and places it in context of the database literature reviewed in this chapter.

Like the standard database architecture defined in the ANSI/SPARC model, the design of a multi-model DBMS separates external and internal levels by means of a conceptual level. But, its architecture takes the divide-and-conquer strategy of query evaluation in relational database systems several steps further. The implementation is separated in three layers: the conceptual, the logical, and the physical layer. Instead of transforming complex object queries directly into operations in a physical algebra, as is common in most database architectures, query evaluation in a multi-model DBMS takes place in several phases. Both the logical and the physical layer can be extended with domain-specific data types and operators.

Each of these layers can be viewed as implementing a complete three-schema architecture with its own data model and query languages. The internal level of the

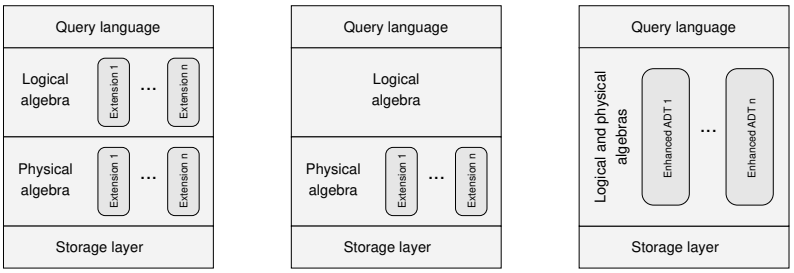


Figure 2.4. The multi-model DBMS architecture next to the extended relational and E-ADT DBMS architectures (from left to right).

conceptual layer corresponds to the external level of the logical layer, and similarly, the internal level of the logical layer is the external level of the physical layer (see also Section 2.10.3). The physical layer is implemented as a traditional DBMS that converts queries against its data model into actions in the file system. But, query evaluation is aware of the expressions in all the layers, which allows each layer to participate in decisions determining the best query plan. The different data models at different layers are used as a design mechanism to tackle query processing at various levels of abstraction, reducing the complexity of the problem.

2.9.3 Discussion

At the first impression, the idea of a multi-model DBMS looks very similar to the way object wrappers provide an object-oriented layer around relational database systems. But, the details matter, which is again explained best using the open implementation principle. An object wrapper considers the relational DBMS (that implements the physical layer of the architecture) as a black box. As discussed before, the object wrapper will have to provide its own query optimizer, as the relational optimizer does not know the strategy that generates the queries at the physical level: a hematoma of duplication. In a multi-model DBMS, there is one query processor, that uses different representations of data and queries at the different levels of its design. But, it has access to the representations at the lower level, and can make its own decisions at different levels of abstraction.

The crucial architectural difference between a multi-model DBMS and other extensible database systems discussed in this chapter is that query processing at the logical layer uses only operators that are provided by the physical layer. This enforces a system-wide physical data model and algebra spanning all extensions, that can be extended if really necessary. This design provides the query processor with transparancy through the layers. In a way, query evaluation can 'look down' from the original request through all layers of the architecture. This design should make it possible for optimizations to also cross the boundaries between (enhanced) ADTs. Therefore, almost all query evaluation can be performed set-at-a-time, and can be optimized using parallelization and pipelining. If logical operations cannot be expressed efficiently in

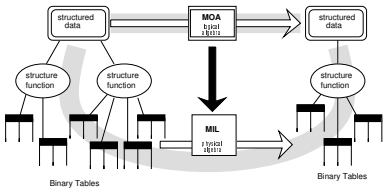


Figure 2.5. Moa query execution by translation to MIL

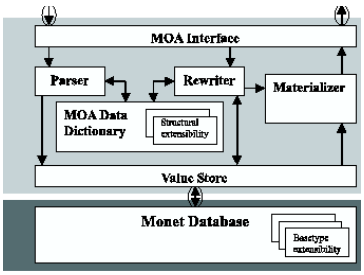


Figure 2.6. The design of the So-Simple DBMS

the available physical algebra, then the physical algebra is first extended with new set-oriented operators.

2.10 THE SO-SIMPLE DBMS

This section presents a prototype implementation of a DBMS based on the multi-model DBMS architecture. It is a simplified model of a complete DBMS, aimed to study efficient query evaluation and domain-specific extensions in a structural object-oriented framework as suggested by Steenhagen. For its simplicity and its emphasis on structural object-orientation, this thesis refers to the prototype as the So-Simple DBMS. It is used as a research vehicle for validation of the multimedia database architecture proposed in this thesis.

The **logical data model**, the data model used in the logical layer, is a complex object model that is queried with a logical algebra. The **physical data model**, used in the physical layer, is the binary relational data model. The mapping between the logical and physical data model is performed in Moa, a generic algebraic framework (see Section 2.10.1). Monet, a binary relational DBMS (see Section 2.10.3) provides the physical level. A *structure function* specifies the correspondence between a structured Moa type and its binary relational representation. Queries formulated against an object schema are mapped into series of algebraic operations in binary relational algebra. Figure 2.5 illustrates this process: the query is a Moa expression on a structure function of binary tables. Its translation is a sequence of operators on the operand tables, that generates result tables, which in turn are operands of another structure function representing the Moa result value.

The So-Simple DBMS implements only these two layers (logical and physical). Thus, in this thesis, the conceptual data model is identical to the logical data model, and users can only express their queries in object algebra. It is anticipated that future revisions of the So-Simple DBMS will support a more powerful data model at the conceptual layer, that supports features such as object identity and type inheritance. Also, full support for an object calculus like OQL should be provided. Because only the logical level has been implemented so far, queries presented in this thesis are always expressed in the logical algebra though.

The architecture of the resulting database system is shown in Figure 2.6. The parser and rewriter manipulate Moa expressions, and generate queries for the physical layer. The materializer operates like a cache between clients and the data in Monet. The Moa kernel and the discussed extensions have been implemented in approximately 5000 lines of Java code. About 3500 lines of these form the generic algebraic framework; the remaining 1500 lines implement the basic extensions used to define a complex object model and its algebra.

2.10.1 Logical data model and algebra

Moa is an extensible algebraic framework. The Moa kernel supports an abstract notion of orthogonal structures. Actual implementations of structures are added to the kernel in separate modules. Core Moa is an abstract framework that allows for the definition of type constructors (which together form a structural object-oriented data model) and operations on types. In the following, the definition of a data model and algebra in an unspecified Moa instantiation is discussed first. Next, the particular instantiation of Moa used in the logical layer of the So-Simple DBMS is treated.

Data model. Moa supports the definition of data models based on the principle of structural object-orientation. Moa assumes a finite set of ADT-style base types. Base values (which are instances of base types) are atomic; their internal structure cannot be accessed, and is only accessible via operations. Base types are implemented at the level of the physical storage system, and therefore allow efficient implementation in a general-purpose programming language.

A structuring primitive, or structure for short, combines known types to create a structured type. Structures can be used recursively. Moa's type system can now be summarized as follows:

- *base type*: τ is a base type if τ is an atomic type at the physical level.
- *structured type*: If $\tau_1, \cdots, \tau_n$ is a, possibly empty, list of types and $\mathcal{T}$ is a structure defined over $\tau_1, \cdots, \tau_n$, then $\mathcal{T}(\tau_1, \cdots, \tau_n)$ is a structured type.

The collection of structures constitutes the data model at the logical level. These structures are mapped by the Moa implementation on the physical system. This mapping provides data independence between the Moa data model and the physical storage system. The logical-to-physical mapping also allows the system to optimize query execution plans.

Algebra. The algebra on the Moa data model consists of the operations defined on the available atomic and structured types. Each atomic type has its own accessor functions. The accessor functions on the atomic types are executed directly in the physical system. Each structure definition comes with operations on the structure.

Instantiation. In the prototype implementation of the So-Simple DBMS, the definition of kernel extensions `atomic`, `tuple`, and `set`[13] instantiate Moa as a logical algebra on complex objects. The base types are provided by the underlying physical

system, Monet. Since Moa structures are orthogonal, the defined data model subsumes the NF^2 data model.

The Moa kernel translates logical operations on structures into efficient physical execution plans. The `set` structure provided in the current implementation specifies operators that are common in query algebras: selection, map, join, semijoin, nest, unnest etcetera. The `tuple` structure implements an operation that extracts an attribute from the tuple. Atomic types simply add their accessor functions to the algebra. Details of the mapping, as well as an evaluation of its performance, are described in [BWK98].

2.10.2 Examples

Let `C` be a value of the following type, which could be used in a data model for the compact disc example of Figure 2.1:

```
SET<
  TUPLE<
    Atomic<str>:  artist,
    Atomic<str>:  title,
    SET< Atomic<str>:  songtitle >:  songs
  >
>;
```

The `select` operation below selects the compact discs from 'PJ Harvey'. The result of a `select` is a value of the same type as the operand. After the selection, the `artist` attribute has been removed from the resulting tuples using a `map` operation.

```
map[TUPLE< %title, %songs >](
  select[%artist = 'PJ Harvey'](  C )
);
```

The next example shows how to select all compact discs with the song 'Helter Skelter'. Although the select in Moa is nested, the expression translates into a set-oriented query plan at the physical level, in which the nesting has been removed.

```
map[TUPLE< %artist, %title >](
  select[ count(select[%songtitle
     = 'Helter Skelter'](%songs)) != 0 ](  C )
);
```

Assume now that `Q` is a value of type `SET< Atomic<str> >`, containing some names of artists. The following `join` operation joins the elements of set `C` with the elements of set `Q`, on equality of the `artist` attribute of `C`:

```
join[%artist, THIS, TUPLE<>](  C, Q );
```

This join uses the `TUPLE` structure to generate the result. Consequently, the type of the result is shown below. If preferred, another `map` operation may remove the

matching `artist` attributes from the result, turning the expression into a natural equijoin.

```
SET<
  TUPLE<
    TUPLE<
      Atomic<str>,
      Atomic<str>,
      SET< Atomic<str> >
    >,
    Atomic<str>
  >
>;
```

2.10.3 Vertical fragmentation and Monet

Supporting a complex object model at the logical level on a binary relational model at the physical level requires vertical fragmentation of the complex object structures.

Copeland and Khoshafian were the first to realize the advantages of a fully decomposed storage model [CK85]. Surrogate keys, maintained by the system, associate separately stored attribute values with corresponding record structures. For performance reasons, they propose storing two copies of each attribute relation, one clustered on the attribute value and one on the surrogate. Under these assumptions, an analytical model demonstrates that a relational DBMS using this storage model can provide efficient query processing, without losing a lot of performance for updates. The advantages of vertical fragmentation are even more promising for the support of different data models. A decomposed storage model seems better at handling set-valued attributes, object identity, the management of inhomogeneity in record structures, and the support of directed graphs and multiple parent relationships (important for type inheritance). Furthermore, decomposed storage has better characteristics for query processing with limited buffer space, is better equiped to take advantage of parallelism and pipelining among multiple operations, and is very suited for a hardware architecture with multiple independent disks.

While Copeland and Khoshafian propose vertical fragmentation as an implementation technique for the internal level of a DBMS, Monet supports a binary relational data model at its conceptual level. Monet is an extensible parallel database kernel that has been developed at the UvA and the CWI since 1994 [BK95, BK99]. The data is stored in Binary Association Tables (BATs). BATs are tables with two columns, called head and tail, respectively, each storing atomic values. Structured data is decomposed over many narrow tables. The system is intended to serve as a backend in various application domains; it has been used succesfully as backend for geographic information systems [BQK96] as well as commercial data mining applications.

Monet allows the *specification* of full decomposition in a declarative manner, without prescribing precisely how the binary tables are stored on disk. Compared to decomposition as a storage model, Monet's physical data independence makes it pos-

sible to take better advantage of the benefits of decomposition discussed in the previous subsection. For example, if Monet 'knows' that a BAT with surrogates in the head and atomic attribute values in the tail has been ordered on its head, the surrogates are not necessarily materialized on disk. Note that this optimization does not violate ordering independence; Monet simply materializes the surrogates again whenever the order of the table might change. Also, storing the table in two ways becomes a matter of physical database design, which can be performed independent of modeling an algorithm as a sequence of set-oriented operations on binary tables.

Monet's design is motivated by two trends. First, the average main memory in workstations gets larger and larger. Therefore, processing should be focused on operations that are performed in main memory instead of on disk; all primitive operations in Monet assume that their data fit in main memory. Second, operating systems evolve towards micro-kernels, i.e. they make part of the OS functionality accessible to applications. A DBMS should therefore not attempt to 'improve' or 'replace' the OS-functionality for memory management. If the tables in Monet get too large for main memory, the database uses memory mapped files. It uses the lower level OS primitives to advice on the buffer management strategy, which is then often delegated to special-purpose memory management hardware.

MIL, the Monet Interface Language, provides a BAT-algebra, consisting of basic operators on bags (or multi-sets), including `select` and `join`, some less standard operators like `semijoin` and `antijoin`, as well as a collection of control structures. The operators perform an additional **dynamic optimization** step before execution. Based on the properties of the operands, the most efficient algorithm is chosen. For instance, the join algorithm may be a `hashjoin`, but also a `mergejoin` that assumes the join columns to be ordered, or a `syncjoin` that assumes the join columns to be identical. When two BATs have an identical head column, they are said to be synchronized. Join operations can be performed very efficiently on synced BATs, because we do not have to compare the values - we know beforehand that the head values in the operands are equal if they are in the same position. This way, the extra cost for re-assembling the vertically fragmented multi-attribute data is reduced significantly, which has been demonstrated in [BWK98] on the TPC-D decision support benchmark.

2.10.4 Extensibility

The So-Simple DBMS is extensible at all levels of its architecture. At the physical level, Monet can be extended with ADTs to provide new data structures and operations, for example to support geographical information systems [BQK96]. Because Moa's base types are the data types available at the physical layer, the logical layer inherits the base type extensibility provided by Monet. An example of using base type extensibility for multimedia retrieval is the definition of the `vector` type, which is used in Section 3.5.

The logical data model and its algebra are also extensible by adapting or defining new Moa structures. The main purpose of extending Moa is the development of *domain-specific* structures. Such a domain-specific structure can improve upon the general-purpose query evaluation process by using its extra domain knowledge to

generate more efficient query plans. Chapter 5 elaborates extensively on this issue, for the integration of information retrieval and databases.

Extensibility at the logical layer is useful in a somewhat different fashion as well, i.e. to provide new *generic* features in the logical data model. An example is adding support to the logical data model for the notion of object identity. The `object` structure can provide a 'reference' to a value of any type. Its implementation uses the (internal) surrogates defined in the mapping from structured types into binary relations, and makes them available at the data model as references to objects in some class extent. The `value` operator replaces a reference with its value, and this value loses its identity: the identity is only valid within the class extent. A reference and its value can only be combined through creation of a new structured type, e.g. using `map[TUPLE<THIS,value(THIS)>](Oids)`. As the notion of object identity is not directly relevant for this thesis, it is not discussed any further.

2.10.5 Discussion

A disadvantage of developing a DBMS based on an unconventional architecture is that it involves a lot of work to create a complete database system. From an academic perspective, however, it is more important to get the blueprint right and study its implications, than to squeeze the final bit of performance out of an architecture that is clearly 'wrong'. Chapter 5 will demonstrate that the design of the So-Simple DBMS is particularly useful for the integration of IR and databases.

A large proportion of the potential of the multi-model DBMS architecture remains still 'hidden under the cover', because the prototype implementation is incomplete. So far, only a portion of the architecture has been implemented and many issues have not been addressed at all. For example, it is not so clear how to combine a query optimization process with the extensible architecture of the So-Simple DBMS; it seems necessary to borrow some ideas from the design of *Predator*, e.g., let Moa structures provide an optimization interface. Other open questions involve the formalization of Moa, efficiency of queries translated from OQL into the logical algebra, and the processing of updates. But, this architecture will benefit more and more from its advantages, as long as the amount of data increases, its structure gets less homogeneous, and the operations performed become more complex, moving into a situation in which facilities enabling scalability start to play a more important role than has been the case so far.

2.11 SUMMARY

This chapter has reviewed the concept of 'database system'. Sections 2.2 to 2.6 revealed the identity of the database approach and the most important characteristics of database systems. Not concurrency, security, recovery, nor any other 'goodie' *really* tells a database system from other software artifacts; databases are different from other programs because they put the data in the spotlights by utilizing data abstraction. First, this provides data independence, which is important to reduce the efforts of writing and maintaining application programs. Second, declarative query languages enable efficient query evaluation, which is probably the best explanation for the success of

relational database systems: it guarantees ease-of-use, without inducing a performance penalty.

Sections 2.7 and 2.8 discussed the open problems in supporting more 'usable' DBMSs that provide more expressive data models without sacrificing efficiency. Object-oriented database systems are easy to use for the development of single-user applications requiring persistency, but lack the notion of a database schema which impedes the sharing of persistent data between different applications. A database architecture that supports structural object-orientation in an algebraic framework seems a promising alternative, but efficient query processing requires still more research. Extensible and object-relational databases support behavioral object-orientation through ADTs, but run into efficiency and scalability problems because their query processors only see blackbox ADTs.

The chapter concludes with the proposal of a new database architecture, that uses different data models at different layers of its implementation. The So-Simple DBMS is introduced, which is a prototype DBMS conforming to the proposed architecture, and is used throughout the remainder of this thesis. It is argued that a multi-model database system is better prepared than competing database architectures for a future in which scalability is the enabling factor to handle the large volumes of data that we digitize and desire access to.

Notes

1. A database management system can also be *special-purpose* software; whether it is general-purpose or special-purpose, a DBMS is a complex software system, that provides database management as a service to other applications.

2. Tsichritzis and Klug describe data independence in database systems as follows: 'Data independence is not the capability to avoid change; it is the capability to reduce the trauma of change' [TK78].

3. Of course, the conceptual schema cannot be reduced whenever some external schema has defined a mapping to the data that is to be removed. In this case, the database administrator decides whether to remove the applications using these external schemas, or leave the conceptual schema as it is. The contribution of logical data independence in this situation is that a potential inconsistency has been detected automatically.

4. Note however that these transformations can also result in increased execution performance, see e.g. the evaluation of Expression 2.3 in Section 2.6.

5. The equivalent SQL expression is `select title from C where artist in Q;`. Here, the nested-loop is harder to spot, because the existential quantification over the equality predicate is 'hidden' by the syntax, but it is of course the same construct.

6. Both solutions (calculus and algebra) return each matching pair of artists twice: $< a, b >$ and $< b, a >$ both occur. Removing these double results would only have obfuscated the example while it would not have affected its purpose; i.e. demonstrate that an algebra expression is generally harder to construct than a calculus expression.

7. This conflict between encapsulation and data abstraction is reflected in software specification methods. Wieringa observes only one *real* difference between structured and object-oriented software specification methods and techniques, lying in the use of DFDs in the former: 'The DFD technique forces the analyst to distinguish, at some level of the aggregation hierarchy, data stores from data processing components and at that level, we get a decomposition that is incompatible with the object-oriented philosopy.' [Wie99].

8. [Kim94] argues convincingly that the ODMG standard is not a standard, but merely a proposal.

9. The term 'object-relational' was coined by Stonebraker.

10. Date and Darwen mention (in the annotated references) that 'we are prepared to entertain the idea of system keys, and such keys might possibly be thought of as (user-visible) 'tuple IDs'.'. Given the explicit use of `ref` and `deref` operators, using references in Informix Universal Server seems rather close to this idea of system keys. If after say a hundred pages of [DD98] its pedantic style becomes just too much to handle, I recommend reading [Cam96] as a relief.

11. In this example, it is assumed that the nested domain $R(S)$ and the domain S are called 'songs' and 'song', respectively.

12. The count-bug refers to the phenomenon that empty groups disappear from aggregates over nested SQL queries, if the DBMS replaces the nested iteration by a join. A 'solution' for relational systems is to use an outer join instead, which produces NULL values that represent the empty sets.

13. More precisely, the `set` structure defines a multi-set (or bag) of elements.

3
REQUIREMENTS ANALYSIS

Tjielp tjielp - tjielp tjielp tjielp
tjielp tjielp tjielp - tjielp tjielp
tjielp tjielp tjielp tjielp tjielp tjielp
tjielp tjielp tjielp

Tjielp
etc.

—Jan Hanlo, *De Mus*

3.1 INTRODUCTION

A fashion designer and a journalist work with high volumes of multimedia data, and they need a flexible storage and retrieval system to cope with their information collections and *especially* with those of their colleagues. In a closed environment, straightforward solutions with manually added descriptions may suffice. But, as soon as a data collection is shared by more users, more powerful tools are needed. In fact, everybody who collects and uses multimedia data is a candidate user of multimedia database systems.

A multimedia database management system (MM-DBMS) is a DBMS that supports 'normal' users at performing their tasks involving multimedia data. It should in the first place provide the DBMS services discussed in the previous chapter, in the sense that its design supports data abstraction and addresses efficient query evaluation. Also, it

should be more than a backend targeted at application developers: sufficiently powerful ad-hoc query facilities must be integrated in its design.

In recent years, many novel software systems that handle multimedia data have been presented in literature as multimedia database systems. This chapter argues that a large proportion of these systems should not be classified as multimedia database systems. On the one hand, many multimedia software systems support retrieval in collections of multimedia data, but do not provide data independence, let alone algebraic intermediate representations that can be optimized. On the other hand, extensible database systems with multimedia extensions support querying insufficiently, and do not even approximate the retrieval functionality provided in the first type of multimedia software systems.

This chapter identifies what requirements must be addressed in a DBMS to qualify as a multimedia DBMS. It is based on work presented in [dVvdVB98]. First, it discusses what distinguishes multimedia data from other data. Section 3.4 focuses on metadata for the access to multimedia objects. The next section discusses ADT support for multimedia search in extensible databases, using the So-Simple DBMS as an example. Shortcomings of the ADT approach to support multimedia in extensible databases are discussed next, and the query formulation problem is introduced. Section 3.6 identifies new requirements for database systems that reduce this problem.

3.2 WHAT IS MULTIMEDIA DATA?

Computer scientists use the term 'multimedia' to refer to anything that is not conventional alphanumerical data. Sometimes, the term is made more explicit by an enumeration of data types, appealing to an intuitive notion of multimedia: image, audio, video, and text. Heller and Martin use the media taxonomy shown in Table 3.1 for the design and evaluation of the role of the different media in multimedia presentations [HM95]. The horizontal dimension illustrates the many aspects of each media type: an image can be a photograph that tells its own story, or an abstract iconic representation that functions as a metaphor to support information already represented otherwise (typically in the text).

3.2.1 Characteristics of multimedia data

Defining multimedia more precisely than by a list of media types is surprisingly difficult. Grosky attempted to define multimedia data by the human activity involved in creation of the data [GFJ98]. But, long before multimedia data is eventually inserted in the database, movies and music have been created by humans as well, usually with specific care to communicate the artist's message. A claim that the semantics of multimedia data are implicit in the data is debatable as well, because the semantics of an employee number in its numeric representation are no less implicit than the semantics of a stop sign in its iconic representation.

This thesis distinguishes multimedia data (including text) from 'traditional data' (usually alphanumeric) from a query perspective:[1]

Definition 1 *The information conveyed by multimedia data may represent anything from the real-world (unlimited UoD), while the information conveyed by traditional*

Table 3.1. The media taxonomy presented in [HM95].

Media type		**Media expression**	
	Elaboration	*Representation*	*Abstraction*
Text	Free text, sentences, paragraphs	Bold, italics, bullets, underlines, headlines, subheads	Shapes, icons
Graphics	Photographs, renderings, scanned images	Blueprints, schematics	Icons
Sound	Speech, audio transcipts	Intensity, tone, inflection	Sound effects
Motion	Raw film footage	Animation, time-lapsed photography	Animated models, highly edited video

data is a symbolic representation of facts restricted to the database's (limited) universe of discourse.

It follows directly from this definition that queries about traditional data are context-dependent, given the universe of discourse of the database. Conversely, queries in a multimedia database can be independent from the context of the applications for which the data has been collected. Not the form in which the data is represented, but the scope of the semantics captured in the data determines whether it is considered multimedia data or not. Although this definition has its own weaknesses, it is particularly useful for the study of multimedia databases: it identifies applications that use specific characteristics of multimedia data, and therefore may impose new requirements on database systems. Using icons to represent male or female in some employee database does not require extra functionality from the DBMS, except possibly an ADT to handle bitmaps; the only 'new' aspect is that the application developer encoded the male/female distinction with a different set of symbols than the more common choice for a boolean or enumeration type. In a multimedia digital library, an application developer cannot elicit all aspects of the data that may be relevant for its users, and therefore the information encoded in the data cannot be represented using a limited set of encodings that is (or could have been) known beforehand.

3.2.2 Multimedia data with a limited context

Not all applications handling multimedia data require functionality that goes beyond standard DBMS services. Sometimes, it makes sense to limit the context of queries that can be processed to a restricted universe of discourse. As an example, consider a video directory service specialized in soccer, like the one described in [Vel98]. Its architecture consists of two subsystems: a video analysis and a video retrieval component. The video analysis subsystem uses a conceptualization of a limited set

of *predefined* objects and events, including 'ball', 'player', 'goal', and 'corner', that are extracted from the data automatically based on techniques discussed in Section 3.4.3. The retrieval component makes the extracted concepts available to the users. As such, the possible queries are limited to the domain knowledge encoded in the analysis component. Because only alphanumeric data is used for retrieval, this application is served perfectly well by a DBMS with a video ADT.

The point made in this thesis is that storing multimedia data, and querying normal data that has been derived from multimedia data, does not turn a database system into a multimedia database system. A real multimedia DBMS should not restrict its users to predefined access patterns. It could assist developers of the analysis component of [Vel98] with the specification of queries that correspond to a predefined conceptualization. But, these predefined conceptualizations only capture the information required to answer a subset of all possible queries about the soccer videos; end-users of a multimedia DBMS should be able to construct queries for unanticipated information needs as well. For example, marketeers of some multinational firm may be interested to see how often their advertisement in the stadium has been shown on television in the last year.[2] Requirements on the support of ad-hoc query functionality are further discussed in Section 3.6. Of course, it may be impossible to formulate and answer a query for an information need, given the digitized data and the current state of automatic analysis techniques. But, the decision whether it is worthwhile to spend more time to try and find relevant objects for an ad-hoc query should be left to the user.

3.2.3 Composite multimedia objects

Multimedia data can be described as a structural composition of atomic objects at different levels of granularity. For example, a news bulletin is a sequence of news items, usually consisting of an announcement, an interview with a reporter at the spot, and some shots about the event that is reported about. Similarly, a book is a sequence of chapters, in which each page may contain text, tables, and images, in which text consists of words, headings, and punctuation, and so on.

Hardman reviews document models for authoring multimedia and hypermedia in her thesis [Har98]. She identifies the following basic components of a hypermedia document model: media items, (media independent) structure, anchors, and presentation specification. This subsection shortly reviews these components, and discusses how these aspects of multimedia data affect a multimedia database.

In Hardman's model, a composite multimedia object is composed of one or more media items. A media item is defined as an amount of data that can be retrieved as one object from a store of data objects, e.g., a piece of text, an image, a video, or a sound fragment. It is an atomic object: a media item cannot be divided in smaller objects without losing its meaning. The granularity of this atomicity is determined by the objects' semantic content, and therefore subjective to the opinion of the creator of the object. The composition of a multimedia object is defined by its media independent structure, which groups media items included in the object into 'sub-objects' which can be manipulated as one entity, and thus can in turn be grouped.

Often, media items can themselves be represented as a structure over atomic component objects: Hardman calls this the media dependent structure of a media item.

Examples include shots and scenes in a video, regions in an image, and sections in a text. An anchor is a reference to such an internal part of a media item. A video anchor can specify a frame or a sequence of frames in a video object, and an image anchor may reference the region in one of those frames that contains an image. Velthausz introduces a similar concept, called pseudo object, for the representation of regions in video frames that contain e.g. a soccer player, or the ball [Vel98]. The difference between anchors and pseudo objects is their intended use: Hardman uses anchors to reuse content in a different context, while Velthausz uses pseudo objects as input for the analysis of multimedia data.

The presentation specification determines the temporal and spatial structure of a multimedia object; e.g., 'the logo occurs in the top-left corner of the screen', or 'the animation starts after the speech has been played'. From a database perspective, management of presentation specifications is orthogonal to the management of multimedia data used in these presentations. In the anticipated use of digital libraries, authoring a presentation will take place after retrieving multimedia objects of interest. Retrieving ready-to-use presentations that comply with some template specification (which can be viewed as a query) is beyond the scope of this thesis.

This thesis further assumes that a data model based on structural object-orientation with appropriate type constructors can represent both the media independent and the media dependent structure of multimedia objects. In general, the influence of the media dependent structure on the meaning of a media item is unknown. Even for text, little is known about the role of its structure in information retrieval. The situation concerning media independent structure is even worse. Therefore, this thesis focuses on the retrieval of media items. The term multimedia object denotes a media item; unless the term composite multimedia object is used explicitly, the media independent structure of multimedia objects is ignored. Mentioning the structure of an object refers to its media dependent structure.

3.3 ACTIVE VERSUS PASSIVE OBJECTS

The key functionality that a multimedia database should offer is access to multimedia information, as illustrated by the user scenarios described in Section 1.2. With respect to access, Bertino et al. classified multimedia objects in two classes: **active objects** and **passive objects** [BCF97]. Active objects really participate in the retrieval process. Users can specify conditions on active objects in the query, referring to either their content (an image with a red sports car) or existence (a web page with an image and a caption that contains the words 'red sports car'). Passive objects just exist in the database, and it is not possible to condition on the content of passive objects. A passive object can only be retrieved through another attribute.

In general, a database system is often used for associative retrieval: as a tool to recollect unknown properties of stored entities that also have some known properties. Codd called this the symmetric exploitation of a relationship [Cod70]. Associative retrieval is only possible for active objects. Imagine a relational database in which the basic data types are passive. Looking up John's phone number from a table with names and phone numbers would force us to check all records sequentially - each time we want to call John. Strange enough, some software systems that are called

multimedia databases view images, audio and video objects as passive objects, that can only be retrieved by associative retrieval. Users can look at picture number 1500, or play the audio stream related to object 120 in WAV format. But, they cannot search for 'patterns like these', or 'interviews about alcohol and driving'.

If a DBMS handles multimedia objects as passive objects, it is not more than a (possibly huge) collection of multimedia data. This clearly does not meet the requirements of the fashion designer or journalist from Section 1.2, nor suffice for other applications of multimedia databases in police investigation, education, movie industry, travel industry, or home shopping (see [Sub98]). The key activity in all these applications involves searching for objects with some particular content, which leads to the main requirement for a MM-DBMS:

Requirement 1 *Multimedia objects can be active objects.*

Unfortunately, the properties of digitized multimedia objects are not as easily checked as the properties of numbers or strings. Applying an exact match on two digitized media objects will only retrieve another object if it is bit-for-bit exactly the same. The question arises why you would search for a digitized object that you use to formulate the query. Occasionally, associative retrieval may succeed using exact match of digitized media; e.g., a police officer may find the name of a criminal, whose photograph has fallen out of a filing cabinet. However, in most practical situations we do not have the exact picture that resides in the database or we would not search for it. Hence, other means are needed to handle multimedia data as active objects.

3.4 METADATA AND CONTENT

The semantics of multimedia data are implicit to the raw media data. Grosky distinguishes between the digitized multimedia data, and content-based metadata, user-recognizable surrogates for these objects which comprise their content [GFJ98]. The latter can be used to infer information regarding the former's content. This thesis proposes the term **content abstraction** for the process of adding content-based metadata to the raw data, in a way to address Requirement 1:

Definition 2 *Content abstraction is the process of describing the content of multimedia objects through metadata, either assigned manually, or extracted (semi-) automatically.*

By analyzing and processing the raw data, semantics can be made explicit to some extent on different abstraction levels, from feature values (Section 3.4.3) to knowledge-based concepts [BKS98]. Other metadata can only be added by human annotators. This section discusses the various types of metadata that are useful for content abstraction.

3.4.1 Syntactic versus semantic content

There exists a gap between the content as perceived by our senses, and the content that has been interpreted by our brain. For video, Hampapur and Jain call these the audiovisual content and the semantic content, respectively [HJ98]:

- *Semantic content*: The message or information conveyed by the video. For example, after watching a news story about a crime, the viewer has acquired

information about several aspects of the crime, be it what the crime was, where it took place, who were victims etcetera. Notice that semantic content may come from different sources: in movies, it has been constructed explicitly by the director, while in a security video this content is just a log of events 'seen' by a camera.

- *Audiovisual content*: The information that depends on the video and audio signal itself. It is oriented to the aural and visual senses, and does not require an understanding of the data.

When we watch a video, our mind manages to derive the semantic content from the audiovisual content. This process requires a significant amount of background knowledge; it cannot be performed automatically independent of the domain.

A similar distinction between two types of content can be made for other media types: images have pure visual content, and sounds can be experienced without knowing what makes that sound, or ever having heard it before. Borrowing terms from linguistics, this thesis refers to the content of a multimedia object at the perceptual level as its **syntactic content**, and to its interpretation in concepts from the real world as its **semantic content**. 'Syntactic' emphasizes that perceptual features are low-level properties that often mean little or nothing to the user in their bare form.

These different types of content cause a problem for manual annotation. Substantial neuropsychological evidence exists that some properties of audiovisual data cannot unambiguously be expressed verbally. In his book, Iaccino reviews psychological research to differences between the two hemispheres of the brain with respect to perception [Iac93]. While the left hemisphere is verbal and analytic, the right hemisphere is nonverbal and holistic. Each hemisphere is specialized for a different kind of thinking or *cognitive style*. Although a verbal-nonverbal dichotomy associated to the two sides of the brain is still considered speculative, the vast amount of research with split-brain patients and people with cerebral lesions reported in [Iac93] shows convincingly that different areas in the brain are responsible for different perceptual processing. The areas of the brain that handle language are not always involved in this processing. Some of the perceptual information is not mediated verbally, and therefore hard to express in words.[3]

3.4.2 Manually added descriptions

Accepting the idea of different cognitive styles leads to the observation that the usage of textual descriptions alone to search the database may always be too restrictive: the user's valuation processes are different from the query evaluation process that has been modeled in the system. Multimedia retrieval has to take into account more than just textual metadata; the metadata has to capture the syntactic content as well. Despite of this objection, the most common approach to content abstraction is to use manually added textual descriptions for metadata. An advantage of manual annotation is that the search component can be independent of the media type of the objects in the database. Exact match using textual descriptions is well understood; when boolean retrieval fails, information retrieval offers an alternative solution.

An obvious disadvantage is that manual indexing is rather expensive if large amounts of data have to be annotated. Apart from the fundamental problem with manual indexing mentioned before, it is also problematic in three other ways:

- different people use different vocabulary to describe the same aspect;
- different people describe different aspects, and a single person describes different aspects in different situations;
- non-verbal aspects of multimedia data cannot be expressed unambiguously with verbal annotation.

First, it is not likely that people describe objects in a standardized manner. Different people select different words to describe the same concepts. For example, one person may describe a picture of 'an evening in the mountains' as 'dark', while another person describes the same picture as 'somber'. Both try to express approximately the same concept, but if the first searches for the picture in the database collected by the other, he will not find the picture although it is in the database. In information retrieval, this is known as the vocabulary problem [FLGD87].

We may partly overcome ambiguity in natural language using thesauri and (semi-) automatic query expansion.[4] The second problem is however not so easily overcome using linguistic techniques. Different people describe different aspects of the picture. A picture classified as 'A dark natural scene' by an archivist, may be associated with 'An evening at Mount Snowdon' by an enthusiastic hiker. Even a single person will describe different aspects of a multimedia object, depending on the specific situation when asked. A hiker might describe the picture with 'dark natural scene' in his office during the week, but can only remember the picture as 'evening in the mountains' when sitting in his living room in the weekend. In psychology, this phenomenon is known as the encoding specificity principle [MJL76].

The common cause underlying these problems is the limited capability of capturing the full semantics of multimedia data in textual descriptions. Expressing the syntactic content is especially problematic, while this can be an important aspect of multimedia information needs (see, for example, Section 3.6.1).

3.4.3 Approximate retrieval

The QBIC (*Query By Image Content*) system [NBE+93] introduced a new type of metadata for querying images as active objects. The key to the retrieval process is *similarity* between objects. The image query is translated into a point query in some multi-dimensional space called the feature space. Features are automatically derived properties from the content of the multimedia objects. The similarity between a query and a database object is estimated using some distance function in the feature space. Instead of retrieving objects that are identical to the query object, objects that are located close to the query object are retrieved. The term approximate retrieval [Fal96] distinguishes this approach from exact retrieval. Because the features are derived from the content of the objects, the approach is also known as content-based retrieval. The interaction with the user in systems based on approximate retrieval takes usually place following the query by example (QBE) query paradigm. Instead of explicitly dealing

Figure 3.1. Image retrieval based on color features

with the features, the user tells the system what kind of objects to search for by giving one or more examples of 'good' objects.

Features typically represent easy-to-calculate properties of the stored objects. An example feature, often used in image retrieval systems, is a measure expressing the color distribution of the image, such as the color histogram. Other possible features are based on the texture and composition of the image; many examples pass in revue in Chapter 7. Approximate retrieval is not unique for image retrieval. In the Musclefish system, it is applied to content-based retrieval of sounds [WBKW96]. Measures based on pitch, energy, and more advanced audio properties span the feature space.

Features mainly represent syntactic content of the stored objects. The syntactic properties used in approximate retrieval hopefully capture some of the *semantic* content of the multimedia object. Unfortunately, we cannot automatically detect semantic characteristics in multimedia objects. We have to work with the syntactic properties that we can calculate.

A well-known example illustrating approximate retrieval and query by example uses the picture of a sunset as query object. For that particular case, retrieval using color histograms works amazingly well. Other 'good' examples include pictures of 'red flowers' and 'divers'. However, it is not so easy to find a feature space that supports a wide range of queries well. Also, it is not always easy to judge why the system found the retrieved objects similar to the query object. For example, Figure 3.1 demonstrates the retrieved objects if one searches in a small database for pictures of red cars. Also images of buildings and waterfalls are retrieved, which are semantically completely different. In color space however, the picture of a car can be very similar to the picture of a building.

If the features have a clear perceptual interpretation, we may choose to let the user move directly through the feature space. The term navigational querying refers to that situation. Navigational querying has been demonstrated for musicians working with a database of musical instruments [EV94]. In the QBIC system, users could directly manipulate the underlying color query. Essentially, it is just another way to use the approximate retrieval approach. But, many features do not have an intuitive perceptual interpretation, so they are usually not exposed to the user.

3.4.4 Social information filtering

Shardanand and Maes introduced **social information filtering**, an indirect approach to help a user find multimedia objects [SM95]. The underlying idea is that the group of all users of a digital library can be divided into subgroups having similar interests. The user's judgements about multimedia objects, for instance movies or compact discs,

also span a vector space. In this vector space, a nearest neighbour algorithm can find judgement vectors that are similar to each other. Next, the items that appear in the vector of a 'similar user', but have not yet been judged by the current user, are recommended to the current user. This technique is now often used in online stores selling books or compact discs. Upon login, the system asks you to judge a selection of compact discs by several artists. This profile of your taste is then used to find people that like the same discs. If most 'similar' people also judged another record highly, the system recommends it to you.

Similarity between user judgements about objects has three major benefits over similarity between the objects themselves. First, it overcomes the problems with identifying suitable features for objects like music and art. Second, it has an inherent ability for serendipitous finds:[5] you find objects that you like, but did not explicitly search for. Finally, the approach implicitly deals with qualitative aspects like style, which would be hardly possible with automatically derived features.

Social information filtering assumes that groups of similar users probably have similar information needs. The validity of this assumption depends on two factors: the size of the user population, and the scope of their judgements. The number of user judgements should be large enough to enable generalization. But, the domain of the judgements should be rather narrow, because users that have similar interest in one domain may be very different in another domain.[6] Technically, it should not be hard to integrate social information filtering with a multimedia database system. Like approximate retrieval, query evaluation processes point queries in multi-dimensional spaces. The difference between both processes comes down to the difference between the space we map objects in, and the distance measure among these objects.

3.5 MULTIMEDIA DATA AND DATABASES

A multimedia object (e.g. a video or a photograph) can be manipulated in a DBMS by addressing the following three aspects in the schema:

- the digitized representation object;
- data abstraction;
- content abstraction.

A multimedia object (existing in the real world) is represented in the computer by its digitized representation object. This thesis ignores the practical problem that one real-world object may have several digitized representations, produced by varying types of hardware, or stored in different data formats of varying quality; instead, a *single* digitized representation object that represents the multimedia object in the computer is assumed.[7] Data abstraction refers to traditional data modeling, representing attributes such as the title of a movie, or the photographer of a picture. Data abstraction is also used to represent the media dependent structure of the multimedia object. Metadata representing the content of the multimedia object forms the content abstraction of the multimedia object. It can be a textual description of the semantic content, but also a feature representation of perceptual content, extracted automatically from the digitized representation object.

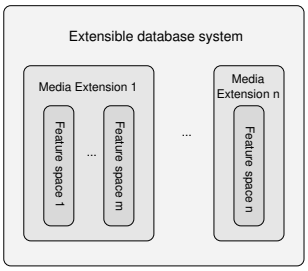


Figure 3.2. Multimedia objects in an extensible database system

In extensible DBMSs, ADTs can support these three aspects of multimedia objects. The general idea underlying multimedia support through ADTs is depicted in Figure 3.2. The DBMS is extended with media extensions $\mathcal{M}_1, \ldots, \mathcal{M}_n$. A media extension supports a single medium, e.g. image or text. It provides support for handling digitized representation objects, as well as distance measures and feature extraction functions for approximate retrieval. For instance, the image media extension might consist of an image datablade providing the base data types to add JPEG images to the type system, and maybe one or more basic retrieval methods like color histogram retrieval. An additional datablade can provide a search space specialized for face retrieval. The feasibility of supporting approximate retrieval with extensible databases has been shown in the Chabot image database. Chabot is implemented on top of the extensible Postgres database [OS95]. Operations defined in the image datablade are used to express conditions for approximate search.

The simplest approach to the management of multimedia data with ADTs stores a multimedia object as a tuple of its digitized representation object and its data abstraction. The media type of the digitized representation object is supported as an atomic data type. The data is viewed and manipulated outside the database, or alternatively, basic operations on these data types have been added as well. Example 1 shows how a video would be selected and played in the So-Simple DBMS, using this straightforward approach to the management of multimedia data.[8]

Example 1

```
map[ play( %video ) ](
  select [ %title = 'Romeo and Juliet' ]( theVideos ))
WITH (
  SET<
    TUPLE<
      video:  Atomic< Video >,
      title:  Atomic< str >
    >
  >:  theVideos
);
```

The schema in the following example demonstrates how media dependent structure of a video object can be represented using structural object-orientation. The video object is now modeled as a structure that consists of the captions or subtitles, the audiotrack, and a series of keyframes summarizing the visual content. The order of the keyframes is ignored in Example 2; if the order of frames is important, a list type constructor can be used instead.

Example 2

```
SET<
  TUPLE<
    subtitles:    Atomic< Text >,
    audiotrack:  Atomic< Audio >,
    keyframes:    SET< Atomic< Image > >
  >
>:  videofragment;
```

Now consider extending the schema with content-based metadata to represent the content of the keyframes. Manual annotation may result in a set of keywords, or some natural language description of the scene. Feature extraction can be performed similar to playing a video in Example 1, calling a feature extraction function instead. Example 3 shows how this content abstraction can be represented in the So-Simple DBMS; instead of a set of images, the schema defines a set of tuples containing the image and its metadata.

Example 3

```
SET<
  TUPLE<
    keyframe:        Atomic< Image >,
    keywords:        SET< Atomic< str > >,
    description:  Atomic< text >,
    colorhist:      Atomic< vector >,
    texture:          Atomic< vector >,
    shape:              Atomic< vector >
  >
>:  keyframes;
```

Approximate retrieval is performed by computing distances to a query vector, and sorting the images in order of their distance to the query object. Example 4 illustrates a query for approximate retrieval using color. Thus, keyframes can now be queried by boolean retrieval using the manual annotations, or by approximate retrieval on color, texture, or shape.

Example 4

```
subrange[1,10](
```

```
  sort[ %theDist ]( LIST<distances> ))
WITH(
  Atomic< Image >:  queryImg,
  map[ TUPLE<
    %keyframe,
    distance( %colorhist,
      compute_colorhist( queryImg ) ):  theDist >
  ]( keyframes ):  distances
);
```

ADTs for multimedia data types allow an extensible DBMS to support querying on textual data and by content. At the first sight, this approach seems to provide the desired database support for multimedia applications. Some problems have been ignored, and it would require more research to confirm that these can be handled in an extensible DBMS without major changes. In particular, the management of media independent structure is a complex problem, because collections of composite multimedia objects can be very inhomogeneous. Handling this situation properly requires data model support for deep inheritance hierarchies, and polymorphic operators, neither of which is supported well in current DBMSs. Ignoring the problems associated with modeling composite objects, integration of approximate retrieval techniques with a (relational) DBMS motivates the development of new algorithms for efficient query processing: the DBMS should support multidimensional index structures, and it should take advantage of the fact that it is often not required to rank all objects. Query processing would benefit significantly from new physical algebra operators that implement the fitness join proposed in [KN98]. Still, these are developments that fit perfectly well in the traditional database architecture, and other application domains (like GIS) would benefit equally well. If this is really all there is to multimedia data management, it does not warrant the recognition of a special class of DBMSs for multimedia . . .

3.6 NEW REQUIREMENTS FOR MULTIMEDIA DATABASES

What most research in multimedia databases tends to overlook (see e.g. [Col94]) is that a database with type extensions is *not* sufficiently usable for multimedia retrieval. Accessing multimedia data puts new requirements on the design of database systems. The examples in this section illustrate the shortcomings of the ADT approach.

3.6.1 The query formulation problem

Part of the journalist's task is to illustrate the article with photos that hopefully attract readers and increase the sales of the magazine or news paper. For news articles, there is not much time to find nice illustrations, and candidate photos are often bought from press agencies, chosing from a rather small collection of recent photos. But, a study of journalists at work, reported in [MS98], made clear that for feature articles, journalists have more freedom than with normal news items. For example, the function of the photo may also be to evoke associations. Also, there is more time to find a 'good' photo.

Imagine a journalist writing an article on 'the effects of the recent economical crisis in Asia'. A journalist usually considers more than one concept for a single illustration task. For the economical crisis example, a possible concept could be a very crowded stock market. Another illustration idea is a photo demonstrating that normal people do not have much money left to spend, for example by showing an empty shopping street in otherwise crowded Hong Kong. In both cases, a photo expressing despair or panic is probably preferred over photos without explicit emotions.[9] Furthermore, constraints like overall page layout may affect the choices made while performing the illustration task.

Assume now that the journalist has access to a video archive of news bulletins originating from various broadcasters. In the archive, time, date, and source are maintained for each news bulletin. The video data itself is modeled with a sequence of keyframes, and a text version of the audio track. The content of the keyframes is indexed using color and texture features. For comparison, a news archive storing similar metadata is described in [HW97].

Searching for 'stock market' in the subtitles may be rather succesful as an initial query. The precision of the results is probably high, meaning that most key-frames with matching subtitles really show stock market scenes. However, the recall may be low: many scenes at stock markets may not have been labelled with an explicit annotation mentioning 'stock market'. Note that this problem will be much worse for the second illustration idea, using 'Hong Kong shopping street' as a text query. But, given a restriction on the location where the video is taken in combination with approximate retrieval using some example frames of empty streets in any city, approximate retrieval techniques may be capable of improving the recall, and really find a shot of an empty shopping street filmed in Hong Kong.

Emotional aspects of the images searched are especially hard to capture in a textual query. Searching for 'despair' in subtitles will probably not retrieve many useful results. These aspects of the illustration task may be captured more easily in terms of feature representations of the images. But, a major problem with the approximate retrieval techniques is starting the process: where should the example objects come from? Querying the features directly is problematic as well. First, a journalist cannot possibly be expected to express a high level concept like 'despair' in a combination of color and texture features. Second, most features lack a clear perceptual interpretation. It is hard to predict what images match constraints like 'circularity $\simeq 0.8$'. The internal representation of the video with content-based metadata should preferably remain invisible for end-users.

3.6.2 Interaction with a multimedia database

So, interaction with a multimedia database faces a major problem that did not exist in the conventional database environment: the users do not know how to formulate their query. As explained in Section 3.4, a multimedia query cannot always be expressed verbally. Nonverbal aspects of multimedia, like emotional and aesthetic values, are hard to capture in words. These values are more easily recognized and compared than described or expressed. The QBE paradigm certainly is a major improvement for such information needs, but, finding an initial example may be just too hard.

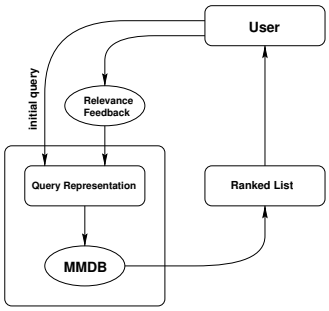


Figure 3.3. The relevance feedback process

Requirement 2 *Query formulation is an interactive process.*

The problem of query formulation is best handled through an interactive search process, see Figure 3.3. Although users cannot express their information need as conditions at the level of features, they *can* tell us which of the retrieved objects are relevant for their internal information need. After an initial query has been processed, the user is asked to judge the retrieved objects. The relevance judgements are used by the DBMS to adjust the query, making it better reflect the user's information need. Processing relevance feedback has been proven effective for both text retrieval and image retrieval (e.g. [HC93] and [STC97]). Querying multimedia needs a discourse and refinement phase for interaction between the user and the database.

3.6.3 Query processing using multiple representations

Current database systems do not provide functionality to capture the user's information need in multiple representations. Conversely, the user views information as a 'gestalt', and each single representation is only a part of it [GJ97]. We cannot expect users to search each representation separately - under the incorrect assumption that they know how to formulate a query to represent their information need - and combine the results by hand. Combination of representations is clearly a task for the database system.

Requirement 3 *Query processing uses multiple content representations.*

Rather than choosing one approach over the other, query processing should use the information from as many (inherently imperfect) ways to describe objects as possible. The usage of multiple representations of multimedia objects is crucial for searching a multimedia database system. Manually added descriptions are not sufficient for multimedia retrieval. Switching to approximate retrieval techniques overcomes some of the problems with textual descriptions, but introduces new problems, because most features have only a syntactic value. As we saw in Figure 3.1, retrieval with color histograms retrieves also waterfalls and buildings instead of cars. Although a single representation of syntactic content is usually not sufficient to capture semantic content, the combination of several represenations may be more succesful. Experiments in image database research, as reported in [MP97], support the hypothesis that

the combination of several feature representations improves the results of retrieval. Early experiences with image retrieval in the Mirror DBMS, discussed in Section 5.5.4, confirm this as well.

3.6.4 Content independence

So far, insufficient usability for the end-user has been the main critique on the ADT approach to multimedia data management. A DBMS that uses ADTs to support content abstraction, which is necessary to address Requirement 1, does not provide *enough* abstraction to enable the average user to query multimedia by content. But, the ADT approach also causes problems for application development.

Recall the main motivation for the development of DBMSs: data independence. Data independence ensures that applications keep running when the internal representation of data changes. Drawing an analogy with data independence helps to pinpoint another problem with the ADT approach: it lacks **content independence**.

Definition 3 *Content independence is achieved when programs that access multimedia data maintained in the DBMS are written independently from the metadata available at some particular moment in time.*

Applications in a multimedia digital library should be able to use the best available techniques to fulfill information needs. But, in current systems, in which there is no separation between the retrieval application and the metadata extraction techniques used internally, applications have to be adapted every time a new technique for content abstraction has been developed and is added to a media extension. For example, when an ADT is added that has a feature space that is highly effective for face recognition, an application programmer has to know about this new feature space, and adapt already existing programs that could benefit from this new feature space. But, the retrieval process modeled in the application does not really change; only the internal representation of content is changed.

Content independence is introduced in this thesis as a notion similar to data independence, but related to the process of content abstraction through metadata. It separates the actual metadata available at some particular moment in time from the query evaluation process. Content independence may be obtained by expressing the part of a query that specifies constraints on the content of the objects only through examples and relevance judgements, without using the metadata explicitly in the query.[10]

Requirement 4 *Query formulation provides content independence.*

Like data independence, a multimedia DBMS that provides content independence increases the value of applications by making them less vulnerable to an ever changing environment. Although retrieval applications would not really break down and stop working without content independence, e.g. when new metadata is added to the schema in an extensible DBMS with ADTs, old applications that do not optimally use the available metadata do not perform as well as new applications; in a commercial setting, this may encourage customers to try their luck with another digital library. Conversely, in a multimedia DBMS that provides content independence, the client applications will

always use all the available metadata; when the DBMS is extended with better multimedia retrieval techniques, its applications benefit automatically from these improved techniques. The following chapter will discuss an approach to providing the proposed content independence.

3.7 SUMMARY

This chapter investigated the combination of multimedia and databases. Using a definition of multimedia that focuses on the difference with normal data, it is explained what applications may require special facilities from a database system. More terminology is introduced, and the first requirement on multimedia DBMSs is proposed: it should be possible to use the multimedia objects as active objects.

Because the semantics of the objects are implicit to the raw media data, content abstraction is necessary to address this requirement. Content abstraction adds metadata to represent the content of the digitized representation objects. It is shown that manually added descriptions cannot express the full semantics of multimedia objects. Therefore, multimedia query processing involves approximate search techniques.

Modern extensible database systems can support these techniques through ADTs. But, an extensible database with multimedia extensions does not provide the functionality that should be provided in a multimedia DBMS, because content abstraction alone is not sufficient. The query formulation problem leads to a different view on query processing than common in the database community. Instead of a one step process with a single query, and the database simply retrieving its matching objects, the interaction between a multimedia database and the user should be a *dialogue*. Query evaluation should iteratively interpret the user's judgements on the results of the previous step, and adapt the initial query such that it will better reflect the observed but unknown information need. It derives database queries against the metadata, using information from the interaction with the user. Since no available technique to handle the objects as active objects captures all semantics of the multimedia data, the DBMS should combine the retrieval results for different representations. Finally, the notion of content independence is defined, and it is proposed that the process of query formulation in multimedia DBMSs should provide such independence.

Notes

1. Admittingly, the term multimedia is somewhat misleading, as it usually denotes *single media* objects. The term multimedia has been chosen over alternatives such as 'media item' to conform to the vocabulary that has become common in database literature.

2. Another example of an unanticipated query requests the video fragment of the Wimbledon final in which a barely dressed lady ran across the main court from a video directory service specialized in tennis. This example is not as absurd as it may seem, given that something similar happened recently at some golf championship, which made it to news bulletins all over the world.

3. Artists have since long been aware of differences in cognitive style. Barrow reports [Bar95] that composer Carl Orff never admitted a boy to the Vienna Boys' Choir if he already knew how to read and write. Apparently, he believed analytical skills block the creative processes needed to develop musical skills. Similarly, the famous composer Mozart asked his wife to read letters aloud during composing. He was convinced that the analytical part of his mind would be distracted by processing the speech and not disturb the creative part making music. At present, Edwards developed a new method for teaching creative drawing, based on these insights in differences between the right and the left part of the brain [Edw93].

4. Even though it may seem 'obvious' that query expansion using thesauri terms reduces the vocabulary problem, a positive effect on retrieval performance has never been demonstrated consistently in large evaluation experiments, such as TREC.

5. Serendipity is the phenomenon of finding valuable or agreeable things not sought for. The word derives from a Persian fairy tale, 'The Three Princes of Serendip'.

6. For example, I like alternative music, but I am not a soccer fan. If the domain used for social information filtering encompasses both music and television, my user profile could match with the profiles of people that like the same music, but also really like soccer; as a result, I would get recommendations for soccer matches on television, which I would not appreciate at all.

7. Notice that this implies ignoring that a different digitized representation may result in different values in feature space for the same object; this can make it impossible to distinguish between identical content (but differences in metadata extracted from different digitized representation objects for the same real world object), and very similar content (but the metadata comes from two different real world objects).

8. Recall that although we prefer a language based on OQL, in the current prototype implementation all queries have to be formulated in Moa.

9. When visiting the Finnish newspaper 'Aamulehti' in Tampere with the Mira working group on evaluation of content-based retrieval methods (see also [PMS+98]), one of the journalists was selecting a photo for an article on floods in Asia. He explained that he was looking for a photo with babies or old people, 'because that catches the eye, and will have more impact on our readers'.

10. In some cases, content-based metadata may be provided as part of the schema too, notably for color distribution, which allows the user to ask explicitly for images with some particular colors in it. But, this does not conflict with the notion of content independence, because this metadata is not used to address the query formulation problem, but as conventional (derived) attributes of the image.

4
CONTENT MANAGEMENT

Gambling is not a vice, it is an expression of our humanness.
We gamble. Some do it at the gaming table, some do not.
You play, you win, you play, you lose. You play.

—Jeanette Winterson, *The Passion*

4.1 INTRODUCTION

The Mirror architecture (presented in Section 4.2) is a blueprint of a multimedia DBMS, which addresses the requirements identified in the previous chapter. This blueprint identifies two new components of a database system that are specific for multimedia DBMSs: the content abstraction component and the retrieval engine. The design of the retrieval engine and its integration with the content abstraction component are the topics of this chapter.

This chapter is organized as follows. Section 4.2 introduces the Mirror architecture. The following section discusses the relationship between multimedia retrieval and IR. The Bayesian view, using probability theory as a logic for plausible inference, is introduced in Section 4.4, as well as its usage in information retrieval. Section 4.5 generalizes the architecture of IR systems to handle differences between text retrieval and multimedia IR. The next two sections deal with evidential reasoning using the inference network retrieval model and an instantiation of this model for multimedia, concluding with a discussion of shortcomings and opportunities for improving the

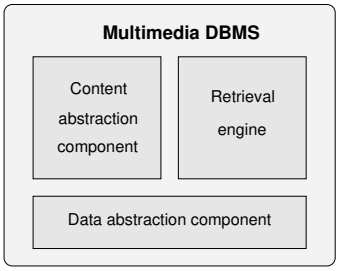


Figure 4.1. Multimedia database architecture

model. Preliminary versions of these ideas have been published in [dVB98b] and [dV98].

4.2 A MULTIMEDIA DBMS ARCHITECTURE

The previous chapter identified four new requirements with respect to the management of content:

- Multimedia objects can be active objects;
- Querying is an interaction process;
- Query processing uses multiple representations;
- Query formulation provides content independence.

By addressing these requirements, a multimedia DBMS combines support for the management of content of multimedia data with traditional database support for the management of the structure of data. Although it is fairly easy to state this goal, meeting it in an actual system is a serious challenge. While the first requirement can be addressed with ADTs in an extensible DBMS, the others demand fundamental changes in database design. The Mirror architecture, presented in Figure 4.1, takes up the gauntlet.[1] Like the ANSI/SPARC architecture, it concerns the definition of interfaces between components, and not a complete instruction for implementation.

The design of a multimedia DBMS is divided into three components. A DBMS like the So-Simple DBMS provides the basic primitives for data abstraction and efficient query processing. The content abstraction component controls the content-based metadata that is available to represent the content of digitized representation objects. This includes descriptions entered by human annotators as well as automatically extracted feature representations. The third component is the retrieval engine, which supports the user with query formulation, using multiple representations for query evaluation. Strict separation between the content abstraction component and the retrieval engine enforces content independence.

Conceptually, the multimedia DBMS operates in two modes (not necessarily mutually exclusive): maintenance and retrieval. In maintenance mode, digitized representation objects can be added or deleted, and the content abstraction component can be

extended with functionality for new types of metadata. In retrieval mode, the retrieval engine interfaces between the internal models of multimedia content and the user. It keeps track of the different metadata that may participate in the retrieval process. Subtasks of the retrieval process may be delegated to the content abstraction component, such as approximate retrieval methods. The retrieval engine 'knows' how to combine evidence from different content representations. It also processes relevance feedback provided by the user, and uses this feedback in the further iterations to refine the initial query. This part of the system takes care of the second and third requirement.

The interface between the user and the DBMS consists of normal data definition and manipulation languages, augmented with some new primitives. First, the data definition language needs a directive that tells the DBMS which entities of multimedia data types that occur in the schema should be treated as active objects. This information is passed to the content abstraction component, which will collect metadata about these entities. As explained before, the retrieval engine maintains a dialogue with the user. The interface should provide directives that declare the start and the end of a query session that is related to some information need. Also, the user needs primitives to specify example objects, and relevance judgements about retrieved objects. Furthermore, the data control language should have a construct for the specification of operations that implement metadata extraction techniques.

The retrieval engine should be based on the theory and techniques developed in information retrieval [dVB98b, dV98]. Its relationship to the content abstraction component and its design and implementation are the research topics of this chapter. Integration of the new components with the underlying DBMS component is discussed in Chapter 5, taking specific care not to affect negatively the set-oriented nature of query evaluation in database systems. The content abstraction component is implemented using ADTs (see Section 3.5). Chapter 6 discusses further implementation issues related to the content abstraction component.

4.3 RELATIONSHIP WITH IR

Van Rijsbergen gives the following definition of information retrieval [vR86]:

Definition 4 *The user expresses his information need in the form of a request for information. Information retrieval is concerned with retrieving those documents that are likely to be relevant to his information need as expressed by his request. It is likely that such a retrieval process will be iterated, since a request is only an imperfect expression of an information need, and the documents retrieved at one point may help in improving the request used in the next iteration.*

Notice the analogy with query evaluation in multimedia databases. Indeed, in their survey of probabilistic models in IR, Crestani et al. remark that a document can be 'any object carrying information' [CLvRC98].

The fundamental concept in IR is relevance, a relationship that may or may not hold between a document and the information need of a user who is searching for information using the IR system. Relevance is usually considered a binary property, i.e. a document is either relevant or not. Although a precise definition has not been accepted, [CLvRC98] describes it as follows: 'if the user wants the document

in question, then we say that the relationship holds'. The set of (binary) relevance judgements is denoted as $\mathcal{R} = \{R, \bar{R}\}$.

An IR system is completely described by its **retrieval model**, which defines (1) document representation, (2) query formulation, and (3) the ranking function [WY95]. If the set of possible document representations is defined as D, and the set of possible queries as Q, then the ranking function $r : Q \times D \rightarrow \mathbb{R}$ maps a query-document pair onto its retrieval status value (RSV).[2] The task of an IR system is to compute this value in response to a query $q \in Q$, and rank each and every document in the collection upon it. The ranking function infers the relevance relationship between documents and queries. It draws conclusions about the relevance of documents observing the available evidence: the document and query representations. Thus, the matching process in *any* retrieval system is a theory of evidential reasoning, whether this theory has been designed explicitely as an inference process, or is only implicitly present in the implementation of the system. Since the representation of the user's information need and the representation of the documents are incomplete and uncertain, (multimedia) retrieval is best described as **plausible reasoning under uncertainty** (in this thesis, plausibility means someone's degree of belief in the truth value of some proposition).

4.4 PLAUSIBLE REASONING AND PROBABILITY THEORY

Plausible reasoning is reasoning about degrees of beliefs [Jay96]. This section first discusses the application of probability theory as a logic for plausible inference; it is based on [HBH88], [Bis95], and [Jay96]. Next, it discusses the use of probability theory in IR.

4.4.1 Probability theory as logic

The conventional, *frequentist* view of probability theory defines probabilities in terms of fractions of a set of observations, and demands that these observations result from a repeatable experiment. The Bayesian formalism is a different school of thought, also known as the *subjectivist* view on probability theory. It allows the interpretation of probability as a degree of belief, such that probability theory can be used as a logic for plausible inference.[3] For example, assume an hypothesis H and background knowledge ξ. If we express our prior belief in H as probability $\Pr(H|\xi)$[4], the Bayesian formalism tells us how to update our belief in H when new evidence e becomes available. Bayesian inference applies Bayes' theorem[5] to determine the posterior probability, allowing us to 'reason backwards':

$$\Pr(H|e,\xi) = \Pr(H|\xi)\frac{\Pr(e|H,\xi)}{\Pr(e|\xi)} \tag{4.1}$$

In 1946, Cox proved that, under some very reasonable assumptions, plausible reasoning is either isomorphic to probability theory, or logically inconsistent.[6] This proof supplies the required mathematical support for subjective probabilities, complementing the more pragmatic 'Dutch book' proofs, which showed that a gambler deviating from probability theory will, in the long run, lose from an opponent that adheres to the rules.

Cox's proof has (or at least it *should* have) a large impact on all applications of reasoning under uncertainty, because it demonstrates that degrees of belief combine just like frequencies. It shows that probability theory is normative for plausible reasoning consistent with common sense, and therefore any alternative that is not isomorphic to probability theory is doomed to produce counter-intuitive results for some models (including certainty factors[7], Dempster-Shafer theory[8], fuzzy logic[9], and other approaches, see e.g. [Par96]). In his (unfinished) book [Jay96], Jaynes gives numerous examples of counter-intuitive conclusions drawn from observed data using other approaches; these include pathologies of so-called 'orthodox statistics', that regardless of Cox's proof still refrain from subjective probabilities, claiming that the data should speak for themselves (*which they cannot*, in any real problem of inference). A possible objection against probability theory as a logic is that it is often more complicated to model ignorance, even though that problem can be solved by choosing an uninformative prior.

In the Bayesian view, computing probabilities is equivalent to performing logical inference, i.e. any computation of a probability can also be viewed as estimating the degree of belief in a Boolean proposition. The product rule (4.2) and the sum rule (4.3) correspond to conjunction and negation, respectively, from which all logic functions can be constructed.

$$\Pr(H_1, H_2|\xi) = \Pr(H_1|\xi)\Pr(H_2|H_1,\xi) = \Pr(H_2|\xi)\Pr(H_1|H_2,\xi) \tag{4.2}$$

$$\Pr(H|\xi) + \Pr(\bar{H}|\xi) = 1 \tag{4.3}$$

In the limit, with $\Pr(H|\xi) \to 0$ or $\Pr(H|\xi) \to 1$, probabilistic inference reduces to Aristotelian deductive logic. But, probability theory provides also a quantitative form of the weak syllogisms, such as abduction. E.g., if ξ includes $H_1 \Rightarrow e$, and we then observe that e is true, then, combining Bayes' theorem (4.1) with $\Pr(e|H_1,\xi) = 1$ (since $H_1 \Rightarrow e$), we get:

$$\Pr(H_1|e,\xi) = \Pr(H_1|\xi) \cdot 1/\Pr(e|\xi)$$

From $\Pr(e|\xi) \leq 1$, it follows that $\Pr(H_1|e,\xi) \geq \Pr(H_1|\xi)$. So, H_1 has become more plausible after observing evidence e; which corresponds to abduction. If the prior probability of observing e is quite unlikely, then H_1 becomes very plausible, a desirable property for diagnosis. Of course, the truth value of H_1 remains uncertain.

A common misunderstanding about the Bayesian inference process (also encountered in [CLvRC98]) is the belief that Bayesian inference cannot handle 'uncertain' evidence, usually combined with the suggestion that some non-Bayesian process known as Jeffrey's conditionalization would be necessary to overcome this 'limitation'. First, the idea that evidence has to be certain in Bayesian inference is a myth [Pea88, p.42]; you simply introduce a mediating variable to model uncertainty in the evidence. Second, Jeffrey's conditionalization is nothing but an incomplete version of the Bayesian rule [Jay96, Chapter 5]. Pearl shows that Jeffrey's conditionalization coincides with Bayesian inference in some particular dependency between the variables, but (obviously) leads to false conclusions if these dependency assumptions are incorrect [Pea88, p.63-70]. Relying on Bayesian inference protects us automatically from drawing such false conclusions.

4.4.2 Bayesian belief networks

Without extra knowledge, Bayesian inference for a problem with n hypotheses forces us to specify $2^n - 1$ probabilities to determine the joint probability distribution, which becomes intractable as soon as n increases. Fortunately, this number can be reduced significantly by specifying assumptions about independencies between different hypotheses.

The most trivial solution (but oversimplifying) is to assume all hypotheses conditionally independent given e, resulting in the 'naive' Bayes classifier [Mit97]. To enable specification of more complex relationships among uncertain beliefs, Pearl developed the formalism of Bayesian belief networks, a graph representation of the independencies between hypotheses [Pea88]. A Bayesian belief network (Bayesian network or belief network for short) is an acyclic directed graph, in which the nodes represent hypotheses (often called random variables) and arcs represent dependencies between hypotheses. The direction of an arc between parent node and child node represents causality. The strength of this causal influence is expressed by a conditional probability.

A belief network encodes the joint probability distribution between all hypotheses efficiently, reducing the number of probabilities to be determined. The complexity of probabilistic inference is reduced by using a very reliable source of information: people's qualitative reasoning about dependencies between hypotheses. Another advantage of the network representation of this distribution is that inference procedures exist to compute the value of any conditional probability in the network given the available evidence, without having to derive a closed form formula for the complete distribution. Although Bayesian inference in arbitrary belief networks is known to be NP-hard [Coo90], probabilistic reasoning is tractable when we apply some restrictions on the network topology. The reader is referred to [Pea88] for more details.

4.4.3 Probability theory and retrieval

Probabilistic information retrieval uses probability theory to implement the ranking function of an IR system. The RSV used for ranking is an estimate of $\Pr(R|q,d)$, the probability of relevance of a document for the user. The 'probability ranking principle' (PRP) states that this ranking is optimal [vR79, p. 113] under the assumption that relevance of a document can be considered in isolation. The problem of applying the PRP is to determine $\Pr(R|q,d)$ accurately enough to approximate the user's real relevance judgment. This is difficult because of the large number of variables involved in the representation of documents in comparison with the small amount of feedback data available about the relevance of documents. Many competing theories can be used to estimate these probabilities ([ZM98] reports on experience with various ranking formulas, and [CLvRC98] gives an overview of the underlying theories), but the results of the current models seem to have reached an upper bound.

Hiemstra approaches the IR problem from a linguistic angle [Hie98]. Explained informally, he constructs a statistical model of the process that created the document (i.e. the author), and estimates $\Pr(R|q,d)$ as the probability that this model would produce the user's query. The proposed model can be rewritten into the well-known

$tf \cdot idf$ product, offering an elegant argument for term weighting in current IR models. A linguistic model provides also an explanation for the performance of an IR system that uses a trigram representation of Swiss-German documents, reported in [GS92]: the measured probabilities are not about the relevance of terms, but about the source producing these terms. Applying the same model to a representation of documents by partial terms (like these trigrams) does not conflict with intuition. But, using a model that estimates the relevance of terms would not make sense, because it is not clear why the relevance of a trigram would be measured just like the relevance of a term.

Van Rijsbergen proposed to estimate $\Pr(R|q,d)$ with $\Pr(d \rightarrow q)$ in a non-classical logic [vR86]. In modal logic, $\Pr(d \rightarrow q)$ can be estimated using a technique known as logical imaging. Logical imaging transfers some of the probability of terms not occuring in the document to the most similar terms that do occur. Experiments on some small test collections indicate that estimating the required probabilities by logical imaging improves performance beyond existing methods, that use $\Pr(d \wedge q)$ [CvR95]. Similarity between terms is defined using the mutual expected information measure, applied over the whole collection. Unfortunately, this approach is based on modal logic, and the question arises whether it would be possible to produce the same results in a Bayesian model. This seems feasible by extending Hiemstra's linguistic model. A model with hidden 'concept' variables, initialized using co-occurence statistics about the complete collection, describes that a concept can be expressed using several similar terms; after observing a document, updating the model implies learning that the author of the document apparently is more likely to chose the terms observed to express that concept, and the posterior probability of that term will increase, similar to the effect described in [CvR95].

Probability theory can also be applied to approximate retrieval techniques. Vasconcelos and Lippman experiment with a Bayesian classification model for approximate retrieval in the image domain, as an alternative for the more popular but ad-hoc techniques using Euclidean and Mahanolabis distance measures [VL99]. They show that color-histogram retrieval is an approximation of the Bayesian classification problem, and confirm experimentally that results improve when using the correct, Bayesian formula. The improvements are even more impressive on a texture retrieval task, for which simple general-purpose DCT features with a Gaussian approximation of the classification problem, succeed in outperforming specialized MRSAR features with Mahanolabis distance, generally accepted as the state of the art in the field.

4.4.4 Belief networks and retrieval

IR evaluation experiments have consistently shown that different retrieval models often retrieve different relevant documents, and that documents retrieved by several methods are usually highly relevant. Therefore, Turtle introduced the formalism of belief networks in IR [Tur91]:

> *Given the availability of a number of representation techniques that capture some of the meaning of a document or information need, our basic premise is that decisions about which documents match an information need should make use of as many of the representations as practical.*

Turtle and Croft model information retrieval using a restricted class of belief networks [TC91]. Fung and Del Favero supplied to the arguments in favour of belief networks in IR that these networks provide an intuitive representation of dependencies between representations [FF95], and explain how to model dependencies between concepts, retaining tractable inference. InQuery, an IR system based on the inference network retrieval model, has proven that the theory can be applied in practice, and it has performed well at several TREC conferences.

On closer investigation, Turtle's belief network model is not 'really' a retrieval model. Rather, it describes a whole range of retrieval models. It has to be instantiated with a fixed structure and probability estimates before it can be used in a retrieval system. For example, they showed in [TC92] how to express the boolean, probabilistic, and vector space retrieval models as inference in Bayesian networks.[10] To distinguish between the generic model and the model's instantiation currently used in InQuery, the latter shall from now on be referred to as the InQuery model.

4.5 DESIGN OF THE RETRIEVAL ENGINE

From an abstract view, the tasks of the retrieval engine in the Mirror architecture are quite similar to the tasks of an information retrieval system. But, the design desiderata of a multimedia database management system differ significantly from those of a special-purpose text retrieval system. After detailing these differences, this section proposes a generalization of the IR system model for multimedia retrieval.

4.5.1 Differences from IR

In a multimedia DBMS, it is not known beforehand what content abstractions of the multimedia objects will be available at run-time. It should be possible to add and remove techniques for the extraction of metadata without having to rewrite the retrieval engine. IR systems have never been designed to deal with such extensibility. Their implementations always assume detailed knowledge about the structure of the indexed documents and the metadata that models the content.

A somewhat related difference between IR and multimedia databases is the number of sources of evidence used in the retrieval process. In IR, only a small number of different sources is considered, e.g. abstract, full text, citations, and sometimes hypertext links. Multimedia retrieval has to combine evidence from *many* different sources. Experiments with the Foureyes learning agent for the Photobook image retrieval system demonstrated advantages of a collection of data-dependent and task-dependent feature spaces over a universal similarity measure defined on a generic feature space [Min96, MP97]. Minka found that different feature spaces capture different aspects of the data; each feature space performs only well at a small set of tasks, on a subset of the data.

Foureyes shows how the retrieval engine may acquire knowledge about the combination and selection of feature spaces, by learning from interaction with the users. But, Foureyes operates in a static environment. The collection of feature spaces in the Mirror architecture changes dynamically whenever new metadata extraction software is added to or removed from the DBMS. This dynamic environment imposes an extra

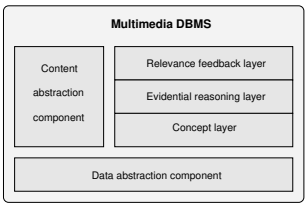


Figure 4.2. The multimedia query processor

problem for the combination of evidence, as it may introduce overlap between content representations when feature spaces model approximately the same aspect. Matching with an object on two different color models has a different implication on the relevance relationship than matching on a color and a texture model. Rather than a carefully selected 'society' of models as envisioned in Foureyes, 'anarchy' seems a more appropriate metaphore.

4.5.2 Multimedia retrieval model

Figure 4.2 shows the Mirror architecture again, this time with a layered design of the retrieval engine. These three layers reflect the aspects of any retrieval model, in subsequently the concept layer (document representation), the evidential reasoning layer (ranking function), and the relevance feedback layer (query formulation).

The concept layer forms the connection between the content abstraction component and the retrieval engine. It defines the basic units representing the content of the multimedia objects. IR literature usually refers to these units as the indexing features; to avoid confusion with the features used in content-based multimedia retrieval, we prefer to call these **concepts**. They can be abstract and intensional, not always corresponding to things from the real world that you can point at (like 'car' or 'tree', see also Section 4.7).

The concepts are input to the evidential reasoning layer, which selects the objects in the database that best match the user's query. This layer has the responsibility to identify the multimedia objects in the database that may fulfill the user's information need as expressed in the query. The evidence is based on the presence or absence of concepts, very similar to traditional IR. The reasoning process combines the evidence from different sources into a single judgement.

Belief networks seem an appropriate foundation for evidential reasoning in the Mirror architecture. The possibility to model dependencies easily, without the need to derive a closed form expression for belief computation, is attractive because we cannot know in advance what feature spaces are going to be available at runtime. Section 4.6 discusses the evidential reasoning layer in more detail.

The relevance feedback layer has two tasks. First, it is responsible for query formulation and adaption. It controls the dialogue between the user and database, analyzing the user's feedback information and changing the query representation such that it (hopefully) better reflects the user's information need. Online processing of relevance feedback is called query-space modification. Second, the relevance feedback

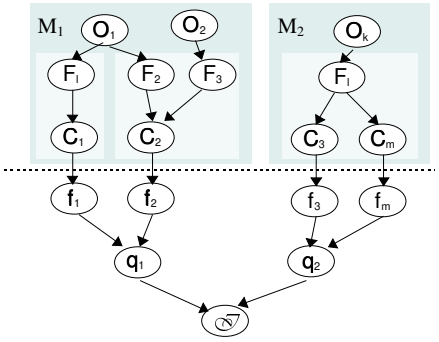


Figure 4.3. A multimedia retrieval model based on Bayesian belief networks

layer maintains a history for offline processing, logging the interaction between users and database. Learning techniques should be applied to improve the representation of the objects in the concept layer, and to identify dependencies between feature spaces. This task is called object-space modification. Although both types of feedback are regarded important, only query-space modification has been applied in the research presented in this thesis.

4.6 EVIDENTIAL REASONING LAYER

This section and the next take the inference network retrieval model as a starting point for the development of a rudimentary multimedia retrieval model (see also [dVB98b, dV98]). Its facilities for the combination of evidence from different sources are the major motivation for chosing this approach. Also, the modular structure of inference networks reflects the extensibility of the Mirror architecture. Instead of developing new theories for IR, which is a goal beyond the scope of this thesis, these sections intend to illustrate how existing theory about inference networks for IR can be applied in the wider context of multimedia retrieval.

4.6.1 Network structure

Figure 4.3 illustrates the general idea behind the use of belief networks for multimedia retrieval. Each base type, e.g. image or audio, has its own media extension $\mathcal{M}_i$. A media extension, depicted as a dark gray box in the figure, manages a collection of content representations $\mathcal{F}_j$, shown as light gray boxes. The nodes in the network represent binary random variables. The top part of the network is called the **object network** and is static for a given data collection. The bottom part, the **query network**, is dynamically created by the relevance feedback layer, based on interaction with the user.

At the roots of the network, we find the object nodes O_i. For now, we will ignore the internal structure of the multimedia objects; all objects are considered atomic. Section 4.7 discusses some alternatives. The objects O_i are connected to their metadata representations of content F_j. The concept nodes C_p represent the concepts

identified in the concept layer. In principle, the concepts may overlap, thus a single representation node may be connected to several concept nodes. Node $\mathcal{I}$ in the query network represents the user's information need. The information need is expressed by the example objects provided by the user in the interaction process. The query nodes q_i model these example objects. The metadata extracted from these objects is represented by the f_j nodes. These nodes are connected to their corresponding concept nodes in the static object network. In the dialogue between database and user, the relevance feedback layer adapts the structure of the query network by adding or removing nodes.

Let us take a closer look at the example instantiation of the network model given in Figure 4.3. Assume that $\mathcal{M}_1$ is an image media extension. It manages feature spaces $\mathcal{F}_1$ for color and $\mathcal{F}_2$ for texture. Image object O_1 has a color feature F_1 and a texture feature F_2. The content layer assigned concept C_1 to color feature F_1, and concept C_2 to texture features F_2 and F_3. The color representation f_1 and texture representation f_2, both extracted from example image q_1, are also assigned to these concepts, hence connected to C_1 and C_2, respectively.

4.6.2 Ranking objects

The inference network computes $\Pr(\mathcal{I}|\mathrm{O}_i)$, which is our belief in fulfilling the user's information need (as expressed in the query network) when this object is presented to the user. In the ranking process, each object O_i is considered in isolation: its node is set to true, and all other nodes to false. This evidence is propagated through the network until it reaches $\mathcal{I}$, when we have computed the desired $\Pr(\mathcal{I}|\mathrm{O}_i)$.

The joint probability distribution encoded in the object network does not depend on the query. In the current model, observing O_i always implies observing its metadata F_j. The feature spaces are assumed independent and equally important. In later revisions of the retrieval model, we may use the conditional probability distribution $\Pr(\mathrm{F}_j|\mathrm{O}_i)$ to represent prior knowledge about how reliably each feature space describes an object. $\Pr(\mathrm{C}_p|\mathrm{F}_j)$ expresses the belief that concept C_p is activated, when feature value F_j is observed. This probability should be estimated during concept assignment. Similarly, $\Pr(\mathrm{f}_j|\mathrm{C}_p)$, specified at the arcs connecting the object network with the query network, describes our belief that feature f_j in query space is described by the concept C_p in object space.

Instead of first computing these probabilities independently, and then propagating these beliefs to the nodes f_j in the query network, the implementation of the inference network retrieval model computes $\Pr(\mathrm{f}_j|\mathrm{O}_i)$ directly. In InQuery, this probability is approximated with term frequency tf, inverse document frequency idf, and default belief α:

$$\Pr(\mathrm{f}_j|\mathrm{O}_i) = \alpha + (1-\alpha) \cdot tf \cdot idf \tag{4.4}$$

Section 4.7 defines a procedure to estimate this probability for multimedia metadata; it should be based on the relative position in a feature space, and the distribution of the metadata of other objects.

4.6.3 Propagation of evidence

To explain the propagation of evidence from the f_i through the query network to $\mathcal{I}$, we introduce a formal description of the inference network adapted from [RM96]. Let x_i be a node in a Bayesian network G, and Γ_{x_i} be the set of parents of this node. Since G is a Bayesian belief network, the influence of Γ_{x_i} on x_i is specified by the conditional probability distribution $\Pr(x_i|\Gamma_{x_i})$. Let the cardinality of Γ_{x_i} be n, and the random variables be binary like in our retrieval model. Then we have to specify $2^n - 1$ different probabilities to describe this conditional distribution. Because the number of concepts related to an object can be high, this is a problem for the computational tractability of the inference. An approximation of the real probability table (also known as link matrix) is needed.

Note that, for a node x_i, the influence of Γ_{x_i} on x_i can be specified by *any* function $F(x_i, \Gamma_{x_i})$ that satisfies:

$$\sum_{y \in \mathrm{Y}} F(y) = 1 \tag{4.5}$$

$$0 \leq \mathrm{F}(y) \leq 1 \tag{4.6}$$

where Y is defined as $x_i \times \Gamma_{x_i}$. In the general theory of belief networks, functions approximating $\Pr(x_i|\Gamma_{x_i})$ have been used to model **causal independence** efficiently: the case when multiple causes contribute independently to a common effect. A famous example is the 'noisy-or' model [Pea88]. In his thesis, Turtle gives closed-form expressions for a limited subclass of functions $F(x_i, \Gamma_{x_i})$, that are useful in IR and can be evaluated in $\mathcal{O}(n)$. Greiff gives a larger class of functions, described by so-called PIC-matrices, for which the evaluation depends on the number of parents that are true but not on their ordering [GCT98]. He first provides an evaluation procedure in $\mathcal{O}(n^2)$, and then gives an algorithm in $\mathcal{O}(n)$ for a subclass of these PIC-matrices. Functions in these classes are 'sum', probabilistic versions of logical operators 'and' and 'or', as well as variations of these usually referred to as 'pnorm-operators'. Together, these functions form InQuery's language for describing the structure of the query network.

Ribeiro claimed to have found an improvement on the InQuery model by reversing the direction of the edges; but, he forgot to take the default beliefs α into account in his (therefore incorrect) comparison [RM96]. His work reports a nice case study of integrating new types of evidence in IR. But, he derives a closed-form formula of the joint probability distribution described by his model, so it is not really an application of the belief network formalism for probabilistic inference in IR.

4.6.4 Some concerns

A weakness of the InQuery model is that it does not explain *why* Equation 4.4 is a good probability estimate. By using this measure, Turtle and Croft claim implicitly that they 'somehow' know that this is the distribution; for, there is no other motivation for this estimate. Indeed, InQuery has used different versions of 4.4 in different publications. The TREC experiments discussed in Section 5.6 of this thesis, have used Hiemstra's ranking in the same network structure, leading to better results; a difference that cannot be explained using the model. Also, InQuery's successful algorithms for relevance

feedback and local context analysis have not been formulated in terms of probabilistic inference, but are processed outside the scope of the Bayesian formalism.

Nevertheless, Turtle and Croft claim advantages of their model over different retrieval models, because of its theoretical foundation in Bayesian belief networks. Due to the simplifications made to the inference procedure and the network structure (trading mathematical correctness for efficiency), it seems however hardly feasible to take advantage of theoretical developments in the more general theory of Bayesian networks without changing (the instantiation of) the retrieval model significantly.

Obviously, an approximation of $\Pr(x_i|\Gamma_{x_i})$ with some function $F(x_i, \Gamma_{x_i})$ is only semantically valid if this function behaves similar to the true probability distribution. Turtle and Croft consider the success of InQuery as 'proof' that these functions model the true probabilistic dependencies between for example the concepts and the document's relevance. This reasoning is clearly flawed. At best, experiments with InQuery can demonstrate that the computed value for $\Pr(\mathcal{I}|\mathrm{O}_i)$ may be interpreted as a reasonable approximation of the true probability of relevance. The distribution captured by *the complete network* apparently reflects some of its desired interpretation in the real world.[11] But, this does not justify the conclusion that the degrees of belief in nodes x_i and their parents have an interpretation as the 'concept probability' regardless of the choice of $F(x_i, \Gamma_{x_i})$.[12] Notice that this objection offers a sound (qualitative) explanation for Greiff's difficulties in experiments with pnorm-operators [GCT98]. Greiff found that it seemed impossible to determine an optimal value for default belief α (cf. Equation 4.4) that worked well in all experiments. But, existence of such a global value for parameter α may not exist. The idea of using a different formula (instead of 4.4) in different nodes warrants further research.

4.6.5 Discussion

The retrieval engine needs a strong theoretical foundation for reasoning under uncertainty, given the complexity of the problems encountered in multimedia retrieval. Despite of some concerns, the idea underlying the inference network retrieval model is very powerful, especially for its ability to flexibly model varying approaches to the combination of evidence from different representations. The proposed model is based on many faulty assumptions, and the approximations used in the inference process violate the rules of correct Bayesian inference. To avoid problems with unclear semantics, using InQuery's operators should be done with care, and the underlying model should be kept in mind. But, direct interpretation of the model encoded by the Bayesian belief network formalism can always be called to rescue, for explaining the good (or bad) performance of some operator in the query network, in combination with some estimate of concept probabilities in the object network.

4.7 INSTANTIATING THE MODEL

A (primitive) multimedia retrieval model in the retrieval engine can now be obtained by instantiation of the abstract model developed in the previous section.

4.7.1 Concepts and probabilities

First, concept assignment should be addressed. Most IR systems use the words occuring in the document as concepts. In text documents, words naturally refer to classes of objects in the real world. For example, the word 'street' occuring in an English text is the same, whether that particular street is located in Cambridge or Oxford. Sometimes, words occuring in the text are first clustered, using stemming algorithms and thesauri. This may alleviate the problems with ambiguity in natural language.

In approximate retrieval techniques for the retrieval of multimedia data, the content representation of objects is a (usually unique) point in multi-dimensional feature space. The feature representation of a street in Cambridge will be different from the representation of a similar street in Oxford. To complicate matters, the representation of one and the same street in two different images will usually be different as well. If the probability distribution of these points is known, then the features can be used directly. However, estimating the distribution on the fly and using it for the retrieval task is not a trivial problem. As a (temporary) solution, the current implementation of the Mirror architecture performs unsupervised **feature clustering** in the concept layer. The simplest approach to clustering defines a grid in the feature space, and interprets each grid-cell as a concept. Alternatively, an unsupervised clustering algorithm, like AutoClass, can be applied [CS95].

Of course, no algorithm will automatically cluster all streets in a single concept. Nor should we expect to identify concepts that only occur in a subset of the streets but in no other classes of objects. But, the assumption underlying the content-based retrieval techniques is that proximity between points in feature space corresponds to some sort of similarity in the real world. Thus, the proximity of the clusters' feature points is likely to reveal an implicit underlying concept that captures some of the semantics of the objects. Hopefully, interaction with the user resolves some of the problems caused by badly formed clusters.

The remaining problem is how to obtain the required probability estimates. The conditional probability $\Pr(\mathrm{C}_p|\mathrm{F}_j)$ is the probability that concept C_p is represented in feature representation F_j of the multimedia object. Ideally, this probability should be estimated using the relative position of the point, and the distribution of all feature points in the cluster. AutoClass can provides such an estimate [CS95], or an alternative like the cluster-based probability model proposed in [PP97] may be considered as well.

Another approach views the concepts as if they are terms in a traditional IR system, and estimates $\Pr(\mathrm{f}_j|\mathrm{O}_i)$ directly like the InQuery model. The current prototype uses this approach, in combination with AutoClass for clustering feature spaces. When applied to image retrieval, the resulting retrieval engine is similar to the Viper image retrieval system [SMMR99], which uses a text retrieval model and a grid clustering of feature space. Chapter 5 discusses some experience with this implementation of the retrieval engine.

4.7.2 Relevance feedback

Learning from the interaction with the user is an important source of evidence for multimedia retrieval. In the current model, query-space modification (or short-term learning) based on the user's relevance feedback is easiest to implement.

Relevance feedback can be processed in several ways. One approach would be to use the observed examples to update the distribution encoded in the network. Unfortunately, it is not obvious how to perform this in the current model, since the InQuery model uses approximations of real Bayesian inference, and, as mentioned before, the probability estimates do not have a theoretical basis in the model. Further research is necessary to investigate this option.

Alternatively, relevance feedback can be used for query expansion. Instead of inference within a fixed query network, a new query network is constructed using statistics derived from the relevant objects. Denoting the set of relevant objects as R_O, selecting the best concepts for expansion can be formulated as selecting the concepts with the highest $\Pr(\mathrm{C}_p|\mathrm{R}_O)$. This probability is estimated with a $tf \cdot idf$ measure, using local tf and idf statistics derived from R_O.

4.7.3 Thesauri and inductive bias

A thesaurus is a set of items plus a set of relations between these items. Often, the relation represented in a thesaurus is similarity between concepts (known as association thesaurus), but thesauri can be used to represent other relationships as well, like generalization or specialization. Van Doorn observed in his Master's thesis that the application of thesauri during query formulation provides an **inductive bias** for multimedia retrieval [vD99]. Inductive bias is the machine learning term for prior knowledge. Any learner needs an inductive bias to generalize beyond the observed training examples [Mit97]. When the number of training examples is large, the need for an inductive bias is low. But, the low-bias approach is not suitable in the interaction with the user, because the sessions acquiring relevance feedback would simply take too long.

Following the design of PhraseFinder [JC94], an association thesaurus can be seen as measuring the belief in a *concept* (instead of in a document) given the query. By representing these concepts as pseudo-documents consisting of their related concepts, using a thesaurus becomes equivalent to the IR problem. Co-occurrence statistics are used to automatically construct pseudo-documents for the similarity relation between concepts. When the thesaurus is constructed temporarily from a part of the collection (for example, using the hightest ranked objects without using prior knowledge), the process is better known as local context analysis (LCA) [XC96].

Thesauri are a flexible approach to extend the Mirror architecture with an inductive bias. Statistical association between captions and pictures has been exploited before in the MARIE project [Row95]. Also, Picard and Minka have experimented with 'visual thesauri', that represent relationships between textual concepts and clusters in various feature spaces [PM95]. In the prototype implementation of an image retrieval system based on the Mirror architecture, thesauri have been used to represent prior knowledge between media extensions $\mathcal{M}_i$. This approach is mainly intended

to bootstrap the query process: a textual description is expanded with concepts from the image extension query, using an automatically constructed association thesaurus. The system recovers from falsely detected associations in the subsequent interaction with the user. Adapting thesauri based on relevance feedback accross sessions would provide a form of object-space modification, or long-term learning. However, in the current implementation of the retrieval engine, the inductive bias does not change over time.

4.8 IMPROVING THE MODEL

The current instantiation of the model is very simple. However, there is much room for further improvements, and getting more out of the theoretical foundation of Bayesian belief networks seems an interesting field for further exploration. This section gives some directions for further improvements.

In the current model, applying Bayesian theory for parameter estimation is complicated, because the model is incomplete: it does not describe how the probability estimates are obtained. Extending the model to make it complete, using an approach similar to Hiemstra's linguistic view on IR, seems the most urgent improvement. An extended model would open the gates to a huge resource of techniques for probabilistic learning (see e.g. [Hec95], [Bun96], and [Kra98]). Learning seems especially useful for the detection of dependencies between feature spaces, by estimating $\Pr(\mathrm{F}_j|\mathrm{O}_i)$. The processing of relevance feedback requires a better foundation as well. Instead of using the (very common) ad-hoc approaches to determine concepts for query expansion, discussed in the previous subsections, this problem should be reformulated and studied as a Bayesian learning task.

Learning dependencies is only one aspect of object-space modification. Object representation is more uncertain than document representation in traditional IR. For text retrieval, word sense disambiguation is one of the most promising approaches to improve document representation. But, in multimedia retrieval, the concepts assigned by the clustering process in the concept layer may in many cases have hardly any interpretation for the user. Detecting such ill-defined concepts is an interesting learning problem, that may improve the retrieval results significantly. However, because multimedia retrieval is so subjective, the amount of learning data from various users is an important factor that should be taken into account before modifying the object-space.

Finally, further revisions of the model should take into account the structural composition of objects from their component objects. For example, the InQuery model has been adapted for the retrieval of compound documents in [Cal94]. Long documents were split in several parts, that were assumed independent. In these experiments, it was found that the best retrieval results were obtained by estimating $\Pr(\mathcal{I}|\mathrm{O}_i)$ as the maximum of the beliefs in the partial documents. But, it is not known how these results may generalize to other collections. Intuitively, it seems unlikely that the maximum of the beliefs would lead to good results in the multimedia case. For instance, a video matching on only the output of a speech recognizer would be retrieved before a video that matches slightly less, but on both the output of a speech recognizer and the subtitles. A lot more research is necessary before modeling the media dependent structure of multimedia objects can improve the retrieval process consistently.

4.9 SUMMARY

The kind of query facilities expected from a multimedia database system are similar to the processes in information retrieval. Based on this observation, a three-layered architecture for the retrieval engine has been proposed, and the tasks of each layer have been explained in detail.

The advantages of the Bayesian network formalism have been discussed in the context of the Mirror architecture, and an adapted form of its application to text retrieval is proposed. The result is a powerful theoretical framework, that serves as a foundation of developing the multimedia retrieval engine. The importance of the model is its direct relationship with Bayesian inference, which should be further exploited in future revisions of the model. Although the assumptions in the current instantiation of the model are not very realistic, there are many approaches to improve the model within the current framework. Despite of its simplicity, the current instantiation of the model is sufficiently advanced to rival existing state-of-the-art multimedia retrieval systems.

Notes

1. Mirror comes from 'Multimedia Information Retrieval Reducing information OveRload'. Also, I find a mirror a nice metaphor for a system that learns what its users look like.

2. In general, the semantics of a retrieval status value are only defined with respect to the one ranking under consideration, and it should not be used for any other purpose than ranking the documents. For example, the RSV for query q_a cannot be compared to the RSV of the same document for another query q_b, nor should the absolute difference between two RSVs be interpreted as an indication of the relative value of the two documents for which it has been computed.

3. The use of probability theory for plausible reasoning dates back to La Place, who wrote: 'The theory of probability is no more than a calculus of good sense' ([Lap02], quote taken from [HBH88]).

4. In general, Bayesians do not always write the background knowledge ξ explicitly in the notation for probabilities, as a degree of belief is by definition conditioned on background knowledge.

5. Although this theorem is generally referred to as Bayes' theorem, Jaynes notes that it had been known long before, e.g. by Bernouilli [Jay96].

6. Cox's original proof has been refined by several later publications, but its result remains known as Cox's theorem. The recent attempt by Halpern to construct a counter-example [Hal99], has been shown incorrect (confirmed by Halpern). In a reaction to this event, Van Horn explains Cox's theorems by presenting a proof based on explicit axioms in [Hor99], which I found much easier to understand.

7. Jaynes remarks about certainty (or confidence) factors: 'the AI theory of confidence factors is stumbling about in a condition more primitive than the probability theory of Cardano, 400 years ago' [Jay90]; he illustrates some counter-intuitive results of plausible reasoning with certainty factors in [Jay91].

8. Pearl explains how Dempster-Shafer theory computes the probability of provability $\Pr(e \models H)$ rather than the conditional probability $\Pr(H|e)$ [Pea88, Chapter 9]. Thus, Dempster-Shafer theory answers questions related to necessity and possibility rather than plausibility, e.g. 'What is the chance that H is necessarily true when observing e?'.

9. Fuzzy logic was not intended for plausible inference, however, it is often (wrongly) applied for that purpose. Van Horn illustrates its inconsistent reasoning in a one-liner: assume proposition A is uncertain, then $0 < (A|X) < 1$, but since $(A \wedge \neg A|X) = \min((A|X), 1 - (A|X))$, and also $(A \wedge \neg A|X) = 0$, so $(A|X)$ must be 0 or 1, which is inconsistent with the assumption of uncertainty [Hor99].

10. Of course, this result is not so surprising given that probability theory is a logic of plausible inference, and belief networks are only a graphical representation of this logic.

11. Maybe I should formulate this more carefully; the current level of recall and precision indicate that the estimates are still far from the *desired* interpretation in the real world. But, the model does not perform significantly worse (nor better) than other IR systems.

12. The flaw in this argument recalls Searle's 'Chinese room'. For, the approximation $F(x_i, \Gamma_{x_i})$ resembles the book with translation rules from English (or any other language, say Dutch) to Chinese. The mistake is to confuse knowing these rules with knowing the Chinese language.

5
THE MIRROR MULTIMEDIA DBMS

"A house can have integrity, just like a person," said Roark,
"and just as seldom."

—Ayn Rand, *The fountainhead*, p. 136

5.1 INTRODUCTION

This chapter presents the Mirror DBMS, the first prototype of a multimedia DBMS addressing the requirements formulated in Chapter 3. Implementing the Mirror architecture requires the integration of IR and databases, a difficult problem that has never been solved completely. Section 5.2 motivates the integration from a technical and functional view. The next section discusses domain-specific structural extensions for the So-Simple DBMS that provides the basic functionality to implement the retrieval engine of the Mirror architecture. After detailing the mapping of these logical structures and their operations to the physical data model in Section 5.4, Section 5.5 discusses the application of these structures in the design and implementation of a prototype image retrieval system. It also describes some qualitative observations about the image retrieval prototype on a small image collection, and also describes some experiments with music retrieval that have been performed. Experience using the Mirror DBMS for TREC is the topic of Section 5.6, as well as some suggestions to improve the efficiency of the current implementation of the belief network structures. The goal of following an algebraic approach for IR is the opportunity to perform query optimization. Section 5.7 discusses the benefits of the Mirror DBMS with this respect. The chapter concludes

with a review of the advantages of the design of the Mirror DBMS, and a comparison to other work. This chapter is based on [dVW99], [dV98], and [dVvDBA99].

5.2 INTEGRATION OF IR AND DATABASES

Chapters 3 and 4 have argued that the integration of IR and databases is a prerequisite for the design of multimedia databases. But, current DBMSs do not sufficiently support searching on content, and, on the other hand, IR systems are not extensible, and cannot handle structured data appropriately. A new type of system is needed, that integrates the management of structure and content. Unfortunately, using database management systems for information retrieval has historically led to impractically slow systems (see also the discussion in section 5.8); the efficient execution of IR techniques seems to require special-purpose software systems.

A characteristic feature of applications that will benefit from integration of IR and databases is the requirement of a combination of content management with the usual manipulation of formatted data; these two aspects of data are often referred to as the **logical structure** and the (implicit) **content structure** of a document [MRT91]. Consider for instance an information request in a digital library for 'recent news bulletins about earthquakes in Southern Europe'. In this example, 'recent' and 'news' refer to attributes of the objects in the library (which are part of their logical structure), while 'earthquake' and 'Southern Europe' refer to the content of those objects (which is part of their content structure). In multimedia digital libraries, such a combination of both aspects plays an important role in the journalist's scenario; the news value of photographs depends in the first place on attributes like date, location, and the identity of people in the picture. Other information systems have requirements that can only be matched with a combination of data retrieval and information retrieval as well: e.g. patient's data in hospital systems, and business reports in office information systems.

Another important reason for integration of IR in databases, that has not widely been recognized yet, is that such integration can help IR researchers to concentrate on retrieval models and reduce the effort of implementation involved with empirical studies. The layered design proposed in the previous chapter separates representation from evidential reasoning and query formulation, which reduces the effort of changing the application logic significantly. The notion of content independence allows using the same applications while experimenting with new theory. Finally, the combination of queries on content with queries on other attributes is a necessary condition for improving the IR process with Mizzaro's different notions of relevance (see [Miz98]).

5.3 IR PROCESSING IN A MULTI-MODEL DBMS

Objections against previous approaches to integrate information retrieval and databases are discussed in Section 5.8. This thesis investigates whether IR and databases can be integrated in a better way, by making the integration *complete*; i.e., neither a layer on top of, nor a black box inside a database system. By extending the structures supported in the So-Simple DBMS with special structures for the retrieval engine, a prototype multimedia DBMS is developed with a much tighter integration than the previous approaches. This prototype system is called the Mirror DBMS. The basic

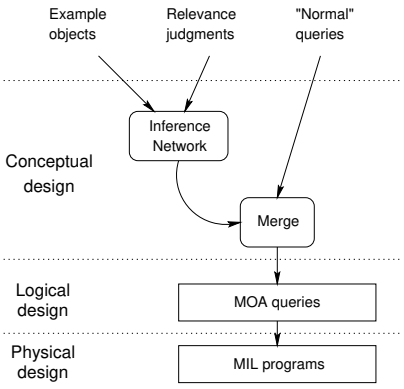


Figure 5.1. Design

assumption is that such a design is better prepared to (eventually) scale up to very large data collections.

Figure 5.1 shows the design of the research prototype. The research has focused on the logical level and physical level. Its main characteristic is the strict separation between the logical and physical databases. This separation provides data independence, and allows for algebraic query optimization in the translation from expressions at the logical level to queries executed in the physical database. Also, parallelisation of the physical algebra is orthogonal to the logical algebra, such that data can be distributed transparently over different database servers by changing only the mapping between the two views.

In the Mirror DBMS, the evidential reasoning process is performed by executing database queries. For this purpose, Moa is extended with structures for components of the inference network. The collection of IR structures extends core Moa with an algebra for IR processing. Operations in this algebra model the propagation of beliefs within a network component. The resulting language allows the specification of many different network topologies, by simply choosing varying operators to combine different sources of evidence. The relevance feedback layer can thus adapt the network structure by simply generating different Moa expressions.

5.3.1 Generic belief network structures

The CONTREP structure covers the object network. It is defined as the content representation of object O_i in feature space $\mathcal{F}$. If an object has metadata representations in several feature spaces, then each combination of object and feature space is modelled in a distinct instantiation of this structure, reflecting the modular composition of the network. Recall that the $\Pr(\mathrm{f}_j|\mathrm{O}_i)$ are estimated directly from the statistical distribution of occurrences of C_p in O_i, and in the collection.[1] Separating the global statistics (representing ξ) from the objects themselves, the object network is described by the statistics about the C_p present in the object O_i. Thus, a CONTREP stores the

connections from node O_i to its associated nodes C_p in $\mathcal{F}$, together with the frequency tf.

Moa is further extended with two other structures, that are used to specify the propagation of evidence through the query network. The `INFNET` structure models a node x_i with its parents Γ_{x_i}. It can be constructed from a set of probabilities, in which each value corresponds to the belief in a node of Γ_{x_i}. The structure defines operators for the class of functions $F(x_i, \Gamma_{x_i})$ that is expressed by PIC-matrices [GCT98]. `DOCNET` is a specialization of `INFNET` that is optimized for the assignment of default beliefs α to nodes that do not occur in the content representation of an object.

5.3.2 Global collection statistics

Another structure has to be introduced before proceeding to examples of the role of these Moa extensions in the various layers of the Mirror architecture. The computation of $\Pr(\mathrm{f}_j|\mathrm{O}_i)$ usually requires global statistics about the document collection. Because several document collections can be managed in a single database, collection statistics are represented explicitly, in a `DCSTAT` structure. A `DCSTAT` can be constructed for a given document collection, using `CONTREP`'s aggregate operator `getGS`.[2] Operation `nidf` calculates the normalized inverse document frequency for the query terms in its operand.

Another reason motivates explicit representation of the global statistics in a separate Moa structure. When a collection is reduced using conditions on the logical structure of documents, in some cases the statistics of the sub-collection should be used for content querying (e.g. when selecting documents that are news items written in English), but in other cases the statistics of the original collection (e.g. when selecting the documents written at University of Twente). An explicit structure for the collection statistics makes both strategies possible. The advantage of this flexibility is most clearly demonstrated in the discussion of query expansion using the user's relevance judgments, later this chapter.

5.3.3 Ranking objects and propagation of evidence

The Moa extensions interact as follows in the computation of $\Pr(\mathcal{I}|\mathrm{O}_i)$. The relevance feedback layer constructs a query network, based on the example objects provided by the user. In the first step of belief computation, `CONTREP`'s operation `getBL` connects the query network to the object network. Its operands are the f_j nodes of the same feature space as the `CONTREP`, and a `DCSTAT` for the global statistics of the feature space. This operation computes estimates of $\Pr(\mathrm{f}_j|\mathrm{O}_i)$, returning a `DOCNET` structure capturing the instantiation of the nodes at the top level of the query network. Evaluation of `getBL` uses the `nidf` operation from its `DCSTAT` operand to obtain the idf values required for estimating $\Pr(\mathrm{f}_j|\mathrm{O}_i)$.

The collected evidence then flows through the network as follows. First, a belief operator $F(q_i, \Gamma_{q_i})$ computes an estimate of $\Pr(q_i|\Gamma_{q_i})$. Next, we repeat constructing an `INFNET` from these estimated probabilities, and computing the belief in the nodes at the next level of the query network, until we reach node $\mathcal{I}$. This procedure computes $\Pr(\mathcal{I}|\mathrm{O}_i)$ using the joint probability distribution described by the inference network.

As an example of this process, assume that `docs` is a set of content representations of text documents, `query` is a collection of query terms (or a collection of tuples of query terms with query weights), and `stats` provides collection statistics. The expression in Example 5 computes $\Pr(\mathcal{I}|\mathrm{O}_i)$ using the InQuery model. Since the `getBL` constructs a `DOCNET`, the inner `map` converts the set of document representations in a set of `DOCNET` structures. The outer map uses the 'sum' belief operator to compute the probability of relevance for each document.

Example 5

```
map[sum(THIS)](
  map[getBL( THIS, query, stats ) ]( docs ));
```

One might have expected the belief calculation of $\Pr(\mathcal{I}|\mathrm{O}_i)$ to be specified completely in the `getBL` operator on structure `CONTREP`, instead of partly in an intermediate structure. Separation of belief computation in two steps has the advantage that the `INFNET` structure permits the representation of more complex network structures, as will be illustrated shortly in an example using compound documents. Furthermore, this approach is better prepared for extensions of the belief network model with the computation of probability estimates specialized for multimedia content. For example, some structure (e.g. `IMGCONTREP`) can construct an `INFNET` for a given query as well, but apply different belief computations than the current IR one. The process of belief propagation encoded in `INFNET` is orthogonal to the computation of these estimates and remains the same.

A network structure consisting of multiple layers can easily represent compound documents. In Example 6, `compounddocs` is a nested collection of content representations, with type `SET<SET<CONTREP>>`; here, each content representation corresponds to a section in the document. This example illustrates the use of the `INFNET` constructor, to express the belief propagation through an extra layer of nodes in the query network. The topology of the inference network specified by this particular query is taken from [Cal94], in which experimental research found that the best results are achieved when a document is ranked by the contribution of its best section.

Example 6

```
map[ max( INFNET<THIS> ) ](
  map[ map[sum(getBL(THIS,query,stats))]( THIS ) ](
    compounddocs ));
```

5.3.4 Combination of information retrieval with data retrieval

Since structures in Moa are orthogonal, schema definitions can represent both the content structure and conventional attributes in a single language. In the following example, let `imgs` be a (manually) categorized photo collection:

```
SET<
  TUPLE<
```

```
        Atomic<Image>:  Photo,
        Atomic<str>:  Category,
        CONTREP: Content
    >
>;
```

Each photo in this collection has been assigned to some category (represented as a string). Using Moa's collection operations, it is straightforward to combine queries on content with queries on category. The Moa expression in Example 7 retrieves holiday images that are most similar to some query image. The `select` retrieves the photos of the 'holiday' category, and the `map` constructs a tuple of the photos with their probability of relevance. To find the ten most relevant holiday photos, simply combine this query with the `LIST` structure and its ordering operations.

Example 7

```
map[TUPLE< THIS.Photo,
    sum(getBL(THIS.Content,query,stats)) >](
  select[ THIS.Category = 'holiday' ]( imgs ));
```

Of course, if a photo can be assigned to more than one category, then the data definition should choose a set-valued attribute for categories, instead of the atomic attribute above. This minor change makes the Moa expression for similar holiday photos more complicated, because it has to select the sets containing the value 'holiday'. Similarly, combining the content of several types of metadata increases the complexity of the expressions as well, because each layer of the inference network corresponds to extra operators to construct the next level of the network and propagate the beliefs. Clearly, these latter examples illustrate the requirement for a higher level language at the conceptual level, as discussed previously in Chapter 2. In the current implementation, however, languages at the conceptual level are not available, and multimedia retrieval applications have to access the Mirror DBMS at the logical level.

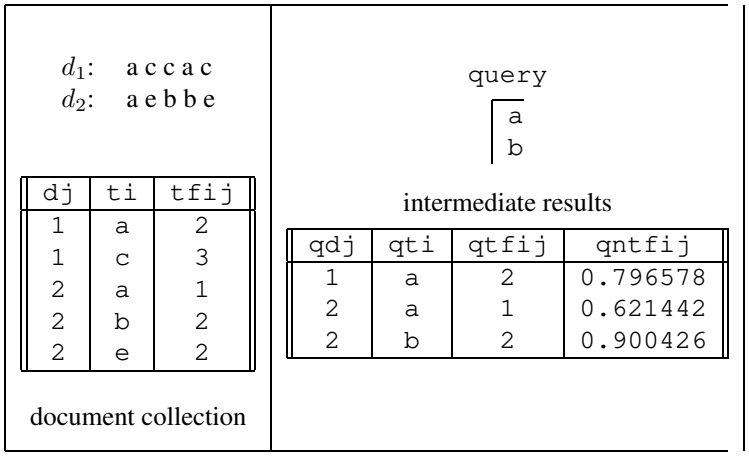
5.4 MAPPING FROM LOGICAL TO PHYSICAL ALGEBRA

The implementation of the Moa structures on Monet requires the mapping of data structures to the binary relational model, and the mapping of operations to sequences of MIL operations. These mappings introduce set-orientation in IR query processing, allowing exploitation of Monet's efficient main-memory query evaluation techniques for the implementation of IR retrieval models.

5.4.1 Flattened representation of content

The representation of the belief network structures at the physical level is termed the *flattened representation* of the content. Table 5.1 shows an example of a flattened collection consisting of two `CONTREP` structures. The Moa structure is stored in Monet as three synchronized BATs `dj`, `ti`, and `tfij`. In the case of text documents, these BATs store the frequency tf_{ij} of term t_i in document d_j, for each term t_i occurring in

Table 5.1. Representation of content in BATs



document d_j; which is a flat representation in the binary relational model of a nested set of tuples with each three attributes. For multimedia objects the t_i are the concepts identified in the concept layer of the retrieval engine.

Computing the probability of relevance of the objects for query q proceeds as follows. First, a table with the query terms is joined with the document terms in `ti` (the result is called `qti`). Next, (using additional joins) the document identifiers and the term frequencies are looked up (`qdj` and `qtfij`). Note that these joins are executed very efficiently, because the Moa structures make sure that the BATs remain synchronized all the time.

Next, the term beliefs are computed with some variant of the popular $tf \cdot idf$ ranking formula. To support these belief computations, Monet's physical algebra has to be extended with new operators. But, because there is no consensus about what belief estimate results in the 'best' retrieval model, many different ranking formulas can be chosen in this step. The various models differ most significantly in the normalization of the term frequencies; some use the sum of term frequencies in a document, others use the maximum, while Hiemstra's model normalizes term frequencies with the sum of all tf_{ij}, the total number of occurrences of terms in the collection.

Monet can be extended with user-defined operators in two ways: define new procedures in MIL, or use ADT extensions (called modules) written in C or C++. Implementing ranking formulas as MIL procedures is very convenient for experimentation; trying a new ranking formula does not require re-compilation. For example, MIL procedure `ntf`, defined in code Example 8, computes the normalized term frequency component tf of InQuery's (original) formula for belief estimation.[3] Dynamic operator binding, another helpful feature of MIL, allows to keep the mapping of the IR process from the logical level to the physical level roughly orthogonal from choosing the probability estimates; in the implementation, switching between InQuery's and Hiemstra's weighting requires only setting one string constant.

Table 5.2. Intermediate tables with and without outer join.

doc	term	belief
d_1	a	0.56
d_1	b	default
d_2	a	0.67
d_2	b	0.82

doc	term	belief
d_1	a	0.56
d_2	a	0.67
d_2	b	0.82

Example 8

```
PROC ntf( tf, maxtf ) := {
  RETURN 0.4 + 0.6*(log10(tf + 0.5)/log10(maxtf + 1.0));
}
```

Thus, after the `qdj`, `qti`, and `qtfij` BATs have been constructed by joining with the query BAT, the user-defined `ntf` operator is used to compute the normalized term frequencies for the terms in `qti`. The necessary document-specific normalization constants are computed using Monet's special set-aggregate versions of the MIL `sum` or `max` operators. Analogously, inverse document frequencies are computed from the collection statistics. As a final step, the term beliefs are computed by combining these idf with the tf; some retrieval models normalize the computed belief with the document length as well.

5.4.2 Probabilistic reasoning in Monet

After the probability estimates for the separate concepts have been calculated, a combined score is computed to express the belief in a document given the query. Monet's modular extension framework is used for the efficient implementation of this belief computation in user-defined operators of module `infnet` (written in C). The class of functions $F(x_i, \Gamma_{x_i})$ studied by Greiff in [GCT98] has been implemented as the probabilistic operators `PICEval` and `PLFPICEval`[4]. Notice that these operators have been implemented both as a normal user-defined operators and as user-defined set-aggregates.

Vasanthakumar et al. [VCC96] have also expressed the computations in the inference retrieval network model with user-defined functions, integrating inference with SQL queries in the DEC Rdb V6.0 relational DBMS. A problem with the implementation of the operators given in [VCC96] is the computation of a full outer join between the terms occurring in the query and the terms occurring in the documents. Thus, the query terms *not* occurring in a document, *are* represented physically in the intermediate results. If the query consists of n_q terms and the collection consists of N_d content representations, then the intermediate result requires space for $n_q \cdot N_d$ records. Because most documents will only match a small portion of the query, the intermediate result is likely to be a long but sparse table, and the storage requirements for this table (in memory) will cause a significant performance bottleneck.[5]

The full outer join is only used to assign default beliefs to the terms that do not occur in the document. A solution without the full outer join is possible when we handle

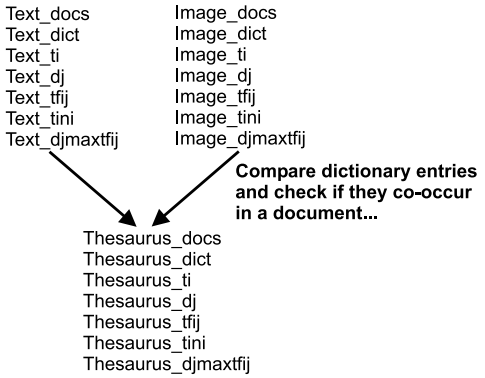


Figure 5.2. Creating a co-occurrence thesaurus from two collections of CONTREPs [vD99].

the assignment of default beliefs locally, *inside* the probabilistic operators. Hereto, the aggregate function `queryPICEval` has been developed, a special version of `PICEval` that takes the query terms as an extra operand. Query weighting (which can be interpreted as attaching uncertain evidence to the query nodes) has also been implemented in a variant of this operator. Instead of inserting default beliefs for all documents beforehand, these operators insert (per document) the default beliefs inside the implementation of the algorithm combining the evidence. Table 5.2 illustrates the difference in processing between the two approaches. To allow an efficient implementation of the `queryPICEval` aggregate, the `dj` and `ti` tables are not only synchronized, but also lexicographically ordered. For some operators (e.g. the probabilistic sum), the default beliefs could be removed from the computation beforehand, and Monet's kernel set-aggregates used instead. But, in general this is not true for other operators; such improvements should be implemented in a query optimizer for the mapping between the logical and physical level.

5.4.3 Thesauri and LCA

Figure 5.2 shows how an association thesaurus can be constructed from two sets of content representations by calculating co-occurrence statistics between the two representations. Building a global thesaurus is a resource-consuming process, that can only be performed offline. But, a *local thesaurus* can be constructed on the fly, for example between the concepts in the query and a small set of content representations. Xu has introduced the term local context analysis to describe this idea of using temporary thesauri for query expansion [XC96].

The Mirror DBMS supports LCA with the `co` operator, an aggregate for collections of `CONTREP` structures. If `contreps` is a collection of content representations, and `query` is a set of query terms, then the following query computes a nested data structure containing co-occurrence statistics:

```
map[TUPLE<THIS,co( contreps, THIS )>]( query );
```

This expression constructs a data structure with three levels of nesting. At the outer level, each tuple stores a query concept, and a set containing a tuple for each object with which it co-occurs. These tuples contain another set of tuples, with a tuple for each concept associated to the query concept. These inner tuples contain the co-occurrence frequency as well. The MIL implementation of this `co` operator does not compute an intermediate sparse matrix of all possible pairs, but investigates only the pairs of concepts that really co-occur.[6] With some simple aggregates, the concepts that co-occur frequently can be selected from the result. Xu's rather complicated formula to rank expansion terms (see [XC96]) has been rewritten in a form such that it is conveniently expressed in a sequence of set-oriented Moa operations on this nested data structure.

5.5 INSTANTIATING THE MODEL

The previous sections have described the mapping of the generic retrieval model into Moa structures, and the implementation of these structures at the physical level. So, the Mirror DBMS provides the primitives for managing multimedia data and its metadata, and for probabilistic inference required during the interaction with the user. This section discusses an instantiation of the model using these facilities, building the retrieval engine of a prototype image retrieval system. Some experience with the prototype regarding the quality of retrieval results is discussed, and some problems are identified. The advantages of using a multi-model DBMS architecture for implementing the multimedia retrieval model are discussed in the concluding sections of this chapter.

5.5.1 An image retrieval application

To validate the ideas about multimedia retrieval proposed in this thesis, the Mirror DBMS has been used as a foundation for a state-of-the-art image retrieval system; aspects of its design are similar in spirit to both the Viper and the FourEyes image retrieval systems. The prototype and its relationship to theories in cognitive science is described in more detail by Van Doorn [vD99].

Content abstraction. The design of the prototype system is shown in Figure 5.3. Obviously, the images can be (partially) annotated with manual descriptions. Also, six different types of content-based metadata are extracted automatically for each image: two color models (RGB and HSB [SB91]) and four texture models (Fractal dimension [CSK93], Gabor Energies [FS89], Grey-Level Co-occurrence Matrix [CH80] and Gaussian Markov Random Fields [CC85]). A combination of color and texture models is used because each describe different aspects of an image. The main reason for choosing these particular texture models was pragmatic: the availability of public domain implementations.[7]

Because the application of learning techniques has not been studied in this thesis, the dependencies between the metadata from various feature spaces are not known. Modelling different nodes F_i for the color and texture features would only make the Moa queries more complex, without changing the inference process performed in the implementation. Therefore, the retrieval model of Chapter 4 has been simplified, such

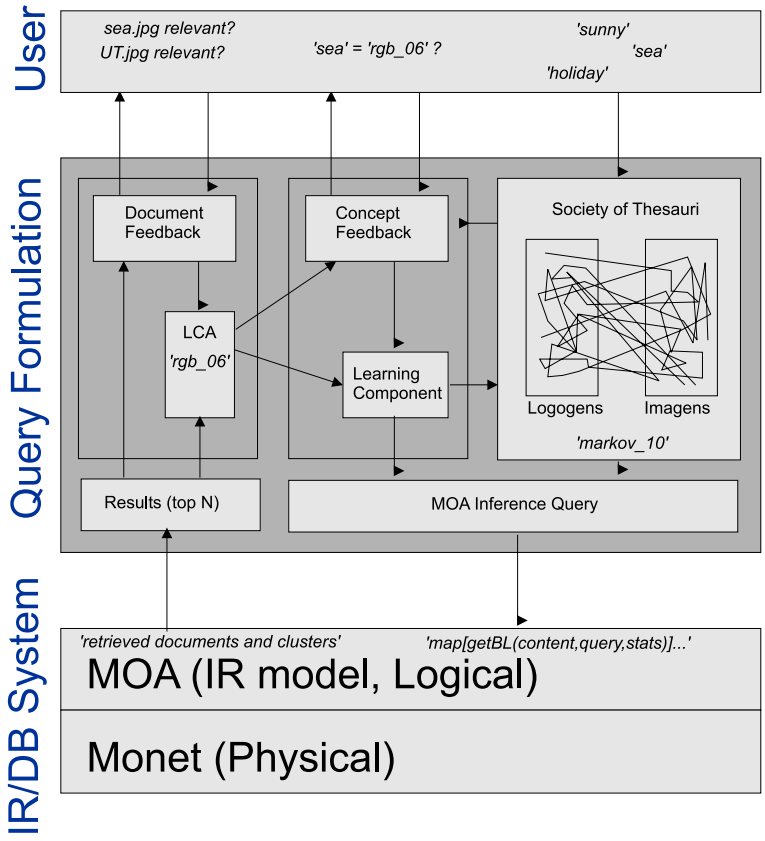


Figure 5.3. Querying image collections using the Mirror prototype [vD99].

that the automatically extracted metadata are used without modelling dependencies between the feature spaces. Thus, the image content is represented by a tuple of two `CONTREP` structures, one for the manual annotations, and one for the content-based metadata.

User interaction. The dialogue between system and user takes place as follows. The user starts a query session with an initial, textual query. This query is passed to a 'society of thesauri', which expands the query concepts with related concepts. The results of the expansion can either be fed back to the user for initial query refinement (via 'concept feedback'), or passed on directly to the evidential reasoning layer. The results of the initial query are passed to the left column. Relevance judgements about the returned documents are introduced in the process through the document feedback unit, and the set of approved objects is examined for query expansion. Alternatively, the highest ranked objects can be assumed relevant (without asking the user), which is known as local context analysis and has performed well in text retrieval. The related concepts discovered in the documents judged relevant by the user are further

handled by the middle column, which deals with concepts. The concepts that can be presented in an intuitive way can be shown to the user for further feedback, however, this requires further research. Another improvement that should be studied is the learning component, which should modify the concept thesauri and/or the document collection, to improve the performance of the system across sessions.

Dual-coding theory. An interesting observation, first made by Van Doorn, is that this retrieval system can be considered an implementation of Paivio's dual-coding theory, a well-known theory in cognitive psychology (see e.g. [EK95]). Paivio claims that human cognition can be explained by the existence of two basically independent, but interconnected, symbolic coding systems. Each symbolic system is specialized in a distinct type of information: the *imagery system* is specialized in processing non-verbal information, and the *verbal system* is specialized in linguistic information. According to Paivio, both coding systems use basic representational units: **logogens** for the verbal system, and **imagens** for the imagery system. The two systems are interconnected by referential links between logogens and imagens.

Paivio's model has been criticized because it does not specify what these representational units are. Although the prototype retrieval application has not been built with the purpose to gain more understanding of human cognition, the (coincidental) similarity between Mirror's retrieval model and Paivio's cognitive model is interesting. In the prototype, the two components of the inference network can be viewed as simple models of the two symbolic systems, in which the stemmed words are the logogens, and the clusters in feature space correspond to the imagens. An association thesaurus constructed between the two collections of `CONTREP`s models referential links between the symbolic systems. Thus, the retrieval model developed for the Mirror architecture can be considered as a (primitive) computational model based on Paivio's dual-coding theory.

5.5.2 Implementing the concept layer

The concept layer constructs the `CONTREP` from metadata extracted from the images. This requires operations for each feature space $\mathcal{F}_j$, to map the metadata into a `CONTREP`. As discussed before, in the multimedia case this implies clustering feature vectors, and in the text case may involve stemming and filtering with a list of stop-words. Another task of the concept layer is to create the `DCSTAT` from the `SET<CONTREP>`.

Figure 5.4 shows the process of creating the representation of the image content in the image retrieval prototype. The first step is scene segmentation or region detection. According to Marr's theory of visual perception (again, [EK95] is a good reference), scene segmentation plays a crucial role in the early stages of low-level perception. A segmentation stage is necessary because most images, like text documents, typically represent multiple concepts (a picture seldomly shows only a clear blue sky or a green field). In the current system, a simple grid segmentation is used, but a high quality region detection algorithm may provide a more natural segmentation.[8] The second step consists of feature extraction on the individual image segments. In the prototype, clustering the set of feature vectors is performed outside the database. Perl scripts

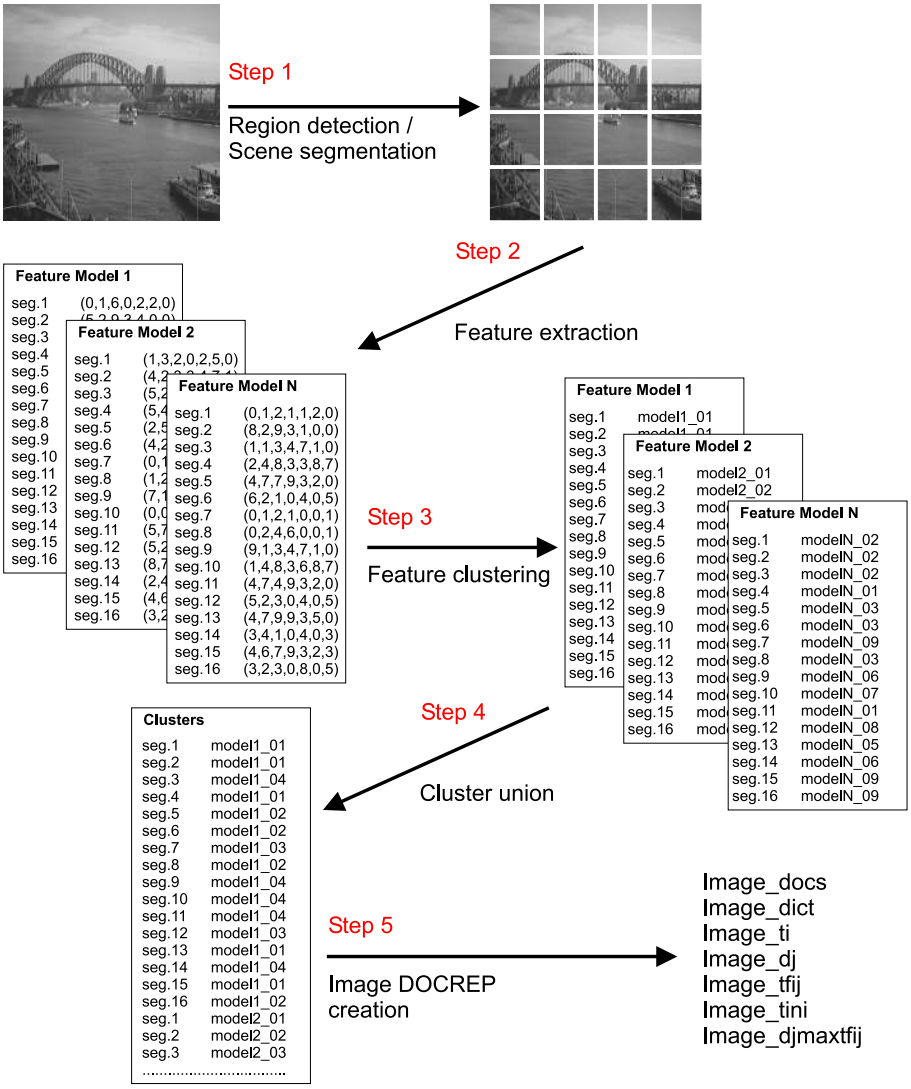


Figure 5.4. Creating the image representation [vD99].

convert the computed feature vectors to AutoClass input files, and, after clustering, convert the AutoClass output files to Monet tables. These tables are then merged and imported into Monet (step four), and transformed into tables for Moa's content representation structure `CONTREP` (step five).

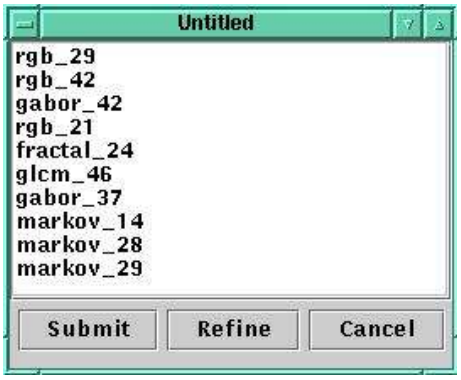


Figure 5.5. Image clusters related to 'street' [vD99].

5.5.3 Query formulation in the relevance feedback layer

Query formulation relies completely on the available content abstractions of the images. The user's information need can only be expressed using example images and relevance judgements; a textual query is considered an incomplete example object, that only refers to the manual annotations of images and not their content.

After the first iteration, the user selects the good and bad images from the ranked list returned by the system. These relevance judgements are used to improve the initial query.[9] With the Moa structures presented before, implementing relevance feedback is trivial. First, the content representations of the relevant images are retrieved, joining the user's feedback information with the presented results from the previous iteration. This results in a (smaller) collection of content representations of good images, from which the global statistics are derived using the `getGS` aggregate. The operators defined on `DCSTAT` then provide the concepts with the highest tf, as well as their $tf \cdot idf$ which are used as weights.

After several iterations, the query process either converges to the best matching images in the collection, or the user will decide that the system cannot find any good images. But, as discussed in Chapter 4, without an inductive bias it may take too many iterations before the query can be derived from the interaction with the user. Automatically constructed association thesauri are therefore used to model prior knowledge. If two concepts taken from the two different `CONTREP` dictionaries co-occur frequently in the same object (e.g. 'sea' and 'fractal_10'), this pair of concepts is associated in some way, and it is added to the thesaurus. The co-occurrence statistics are managed in another `CONTREP` structure, like a multimedia equivalent of PhraseFinder. The thesaurus is used for query expansion with the same Moa expression as in Example 5, using the thesaurus as a set of pseudo-documents containing information about the concepts.

Figure 5.6. Top 10 for the query 'tree, trees, forest' [vD99].

5.5.4 Some observations about the image retrieval prototype

Some small-scale experiments have been performed, to illustrate the weaknesses of the current retrieval model and stress the importance of interactive learning from the user. But, the results of these experiments are presented modestly; for, this research has mainly focused on developing an approach to integrate content-based retrieval techniques with databases, and not on building 'the best retrieval system of this moment'. A discussion of problems with the evaluation of multimedia retrieval is deferred to Chapter 7. Addressing evaluation seriously has been impossible in the scope of this thesis, partially because the Mirror DBMS is still too experimental for real applications with real users.

The image collection consisted of 99 images, and is known as the 'BT collection' [Min96]. It is small, but still challenging because of its diversity (it ranges from faces to cities to landscapes), and the small amount of training data for thesaurus construction. The images have been annotated manually, and the annotations are used to construct a co-occurrence thesaurus. For example, Figure 5.5 shows the clusters co-occurring frequently with the text concept 'street' (obviously, this representation of the image clusters is not intended to be shown to end-users).

Figure 5.6 shows the response for the textual query consisting of the concepts 'tree', 'trees', and 'forest', expanded using the automatically constructed thesaurus. By closer examination of the query formulation process, two classes of errors can be recognized in the concept layer: clusters without an intuitive perceptual interpretation, and clusters with wrong labels in the thesaurus.

In the first type of error, AutoClass finds some clusters of little semantic value, like the cluster occurring in the images shown in Figure 5.7. Because it occurs in images annotated with 'tree', 'trees', and 'forest', the process constructing the thesaurus erroneously concludes that image concept `fractal_23` is related to these text concepts.

If several images have been annotated manually with 'tree' and 'building', co-occurrence data can estimate an association between image clusters that represent some visual aspects of buildings and the word 'tree'. An example of this type of mistake can be found in Figure 5.8.

Figure 5.7. Images containing cluster '`fractal_23`' [vD99].

Figure 5.8. Images containing cluster '`gabor_20`' [vD99].

In other cases though, the low-level image concepts detected in the concept layer correlate nicely with some high-level concept like 'leaves'. As an example, consider cluster `glcm_47` for which the images are shown in Figure 5.9.

Figure 5.9. Images containing cluster '`glcm_47`' [vD99].

The main motivation behind the interactive query paradigm is that the user's relevance judgements allow the system to recover from the unavoidable errors in the concept layer, and improve the query by removing image concepts like `gabor_20` and `fractal_23` and adding concepts such as `glcm_47`. The first impression of relevance feedback is that it really improves the content-based query processing in the prototype system. For example, if the first and fourth image of Figure 5.6 are given as positive examples, the system returns the images shown in Figure 5.10. Also, it has proven possible to retrieve images with faces by giving positive relevance feedback on portaits, even though the current implementation does not have special features for this purpose. But, from such a small collection, no strong conclusions can be made yet about the current instantiation of the retrieval model.

Figure 5.10. The top 6 after relevance feedback [vD99].

5.5.5 An experiment with music retrieval

As stressed in Chapter 3, emotional and aesthetic values play an important role in the user's evaluation process. Because subjective judgments seem especially important when people compare music fragments, the multimedia query processor has also been tested on a content representation of music objects. Notice that the similarity between two fragments is assumed to be defined by the overall 'sound' of the music, and not the melody or lyrics.

Data set **Symbol-1**, created in cooperation with the Dutch company 'Symbol Automatisering', consists of 287 songs. Symbol Automatisering develops tools supporting the creation of radio programs and selecting background music in bars or restaurants; for their users, the similarity between pieces of music is indeed defined by a similar overall 'sound'. Domain experts of Symbol Automatisering have manually classified these songs into six main categories: rock, house, alternative, easy listening, dance, and classical.

The extraction of metadata is based on the Musclefish feature vectors [WBKW96], which have been extended with the output of a simple rhythm indicator, based on peaks in the autocorrelation function of the lowest parts of the frequency domain. The design of this feature extractor has been presented by Oortwijn in his Master's thesis [Oor98]. Between one and two minutes have been sampled of each song, that were segmented into fragments of 5 seconds each. The result is a data collection of 3363 fragments, from which content-based metadata has been extracted. Feature clustering with AutoClass identified 53 different clusters; each feature vector has been assigned the concept node according to the cluster with the highest probability. A song has then been modelled as a collection of these concepts. Like in the image retrieval prototype, this representation of songs has been treated as if they were text documents in which the concepts are the words.

Query formulation on this music collection depends completely on the interaction with the user. A query has been constructed from the concepts that occurred most frequently in half of the songs belonging to a category, simulating a significant amount of online relevance feedback. This query has been entered, to study if it retrieves other songs of the same category. Of the top 20 songs for the query based on 'rock', 15 had also been classified manually as rock. Of the other 5 songs, only 2 clearly do not

belong in the rock category. With the 'classical' and 'house' songs, hardly any misses were found. Results for the category 'alternative' were however hardly better than chance. Maybe this can partly be explained because the 'alternative' category is not so well defined (as confirmed by Symbol); but, only more research with real users may provide an answer to this question.

The resulting system cannot be used for automatic song classification. But, it seems indeed possible to interactively retrieve groups of similar songs, particularly for well defined categories. An improved version of this retrieval system could serve well as the backend of a set of mobile clients: a portable radio with a relevance feedback button, that learns your personal taste of music.

5.5.6 Discussion

This section has described an example of an advanced image retrieval system, based on the functionality provided by Moa's IR structures available in the Mirror DBMS. The current implementation should not be regarded as 'the' image retrieval system supported by the Mirror DBMS; there are numerous ways to improve the prototype. The prototype is not intended to be more than a demonstration of the type of functionality that can be specified with some fairly simple Moa expressions. The main contribution of the Mirror DBMS is that improvements can be specified by simply changing the sequence of Moa expressions. As a result, it is not much trouble to add new feature models, use different clustering algorithms, or apply different techniques for query expansion and query modification.

The techniques for automatic content extraction and learning from the interaction are still too primitive to expect a system like this prototype to be very useful in realistic digital library applications. But, the embedding of content management techniques in a DBMS makes it much easier to integrate more domain knowledge into multimedia applications. While exact data retrieval can be used to prune a large collection into a much smaller set of about one thousand potentially relevant multimedia objects, iterative content-based querying can assist the user with browsing this intermediate result.

In future revisions of the prototype, dependencies between feature spaces should be taken into account, and learning techniques should be used to improve both the object-space and the automatically constructed thesauri. Grid segmentation of the images is too trivial, and should be replaced with better region detection and segmentation algorithms. Also, a pyramid approach to segmentation would improve the quality of the content abstractions, i.e. segmentation should be done at various levels of detail.

Probably, the most significant improvements can be made by modelling the interaction with the user more precisely. At least three possibilities should be investigated for improving interaction with the user. First, binary relevance judgements are not very informative about the user's information need. Instead, users should be encouraged to group the images into similar clusters, and judge these groups for their relevance. Also, the user should be allowed to judge the relevance of segments of images, instead of the complete images. Finally, the user should be asked to confirm hypotheses made in the evidential reasoning layer. As an example of this idea, the PicHunter system provides a good example how a (Bayesian) model of the interaction with the user

can steer the dialogue between the user and the system, and make it converge more efficiently [CMMY98].

5.6 EXPERIENCE WITH TREC

The previous section focused mainly on the functionality supported by the Mirror DBMS. This section presents evidence that the system is a suitable experimentation platform for IR research,[10] and discusses the execution performance obtained with the current mapping of Moa's IR structures into binary relations.

5.6.1 What is TREC?

The most widely used benchmark for information retrieval systems is the TREC test collection.[11] TREC is a series of conferences that provides the research community with a platform for large-scale evaluation of text retrieval methodologies (see e.g. [VH97]). While the TREC collection is large and models a real-world problem, the main target is the effectiveness of a retrieval system rather than efficiency. A system that 'performs well' at TREC retrieves the best documents, and does not necessarily retrieve these documents most efficiently.

The data set consists of several sub-collections of varying size. TREC-6 uses five sub-collections, and for TREC-8 one of these sub-collections has been removed. Each conference, relevance judgements are created for 50 queries (called *topics*), which provide a ground truth for evaluating effectiveness. TREC defines a document relevant if some piece of information contained in the document is relevant for writing a report on the subject of the topic. Because the collection is too large to manually evaluate the relevance of *all* documents, a pooling technique is used. The top ranked documents of the participating systems (fifty-six groups participated in TREC-7) are gathered in the pool, and this pool is assumed to contain all relevant documents. So, a new retrieval system using old TREC relevance assessments may retrieve documents for a topic that have never been judged, because none of the other systems had retrieved these documents before; still, these are regarded unrelevant in the evaluation. Although the pooling technique has been criticized for this reason, the TREC collection is the best option for laboratory experiments.

5.6.2 Experiments with TREC-6 and TREC-8

An evaluation run processes 50 topics in batch, but the client interfaces of the Mirror DBMS have been designed for interactive sessions with an end-user. Also, transferring the data from Monet to the Moa client has been implemented with a lot of overhead. Furthermore, optimizations such as using materialized views are not performed in the current Moa rewriter. These minor flaws would have inferred an unfair performance penalty to the evaluation of the architecture, and made logging the results rather cumbersome. Therefore, as a (temporary) solution, the MIL program generated by the Moa rewriter has been manually edited to loop over the 50 topics, log the computed ranking for each topic, and use two additional tables, one with precomputed normalized inverse document frequencies (a materialized view), and one with the document-specific constants for normalizing the term frequencies.

The machine on which the experiments have been performed is a Sun Ultra 4 with 1 Gb of main memory, running SunOS 5.5.1. The machine is not a dedicated server, but shared with some other research groups as a 'compute server'. Monet effectively claims one processor completely while indexing the collection, or processing the fifty topics on each of the sub-collections. The division of the complete collection in five sub-collections (as it comes on different compact discs) is maintained. The topics are first run in each sub-collection, and the intermediate results are merged. Depending on the size of the sub-collection, estimating the top 1000 ranking takes between 20 seconds and two minutes per topic. How to further improve this execution performance is discussed in the next subsection. Some runs have used global statistics of the merged collection, other runs use local statistics of the particular sub-collection in which the topic is evaluated.

Preparation of the five sub-collections takes about six hours in total. Computing the `tfij` table is performed using the module for crosstables. But, the `CTgroup` operation allocates all available memory, and eventually crashes the DBMS because it cannot get more, if it is run on the complete set of documents of any but the smallest sub-collection.[12] Therefore, the indexing scripts run on fragments of the sub-collections at a time, and frequently write intermediate results to disk, obviously slowing down the process more than necessary.

The importance of measuring effectiveness can be illustrated from the experience with tuning some parameters of the algorithms implemented in the Mirror DBMS on the TREC-6 topics. Using Hiemstra's ranking, the baseline results on the TREC-6 are quite impressive. Based on its success on InQuery at previous TREC conferences, a significant improvement was expected by using topics expanded with LCA. Also, investigating the expansion terms, LCA seemed to do a good job. For example, on topic 311 (which is about industrial espionage), it finds terms like 'spy', 'intelligence', and 'counterintelligence', and from the financial times sub-collection it even identifies 'Opel', 'Volkswagen', and 'Lopez' as relevant terms. But, instead of improving the effectiveness of retrieval, the measured performance turned out to have degraded. Some tweaking of the parameters, reducing the weights of expansion terms and using fewer of them, the performance improved upon the baseline, but only slightly; on the runs submitted for TREC-8, it degraded performance a little bit.

A possible explanation for these disappointing results is that the algorithm has been applied to documents instead of passages (as done in [XC96]), and the TREC collection itself was used to find expansion terms instead of another, larger collection. But, another possibility is that Hiemstra's weighting provides such a high baseline, that it is very hard to improve upon. A comparison between these results and InQuery's TREC-6 report (in [VH97]) favours the latter explanation, because the performance of the Mirror DBMS with Hiemstra's weighting scheme, without LCA, was almost as good as InQuery's performance after using LCA. With LCA, Hiemstra's weighting performed better on all reported precision and recall points, except for the precision at twenty retrieved documents, at which InQuery performed slightly better.

5.6.3 The road ahead

The execution performance of the Mirror DBMS on TREC is clearly better than a naive (nested-loop) implementation in any imperative programming language, but, the obtained efficiency is not fast enough to beat the better stand-alone IR systems that participate in TREC. Compared to the techniques used in systems like InQuery (see [Bro95]), the current mapping between the logical and physical level is too straightforward: it does not use inverted files, has not fragmented the terms using their document frequency, and it ranks all documents even if only the beliefs for the top 1000 are used. Also, Monet should make it relatively easy to take advantage of parallelism in modern SMP workstations.

The merits of some possible improvements can only be evaluated experimentally. For example, it is not so clear beforehand whether inverted files are really the way to go. Query processing with inverted files requires merging the inverted lists before beliefs can be computed, which is hard to perform without trashing the memory caches frequently; which has been shown a significant performance bottleneck on modern system architectures (see e.g. [BMK99] for experiments demonstrating this for Monet).

Without experiments, much improvement can be expected from fragmentation of the document representation BATs based on the document frequency, in combination with the 'unsafe' techniques for ranking reported in [Bro95] (called unsafe because they may affect the order of the ranking). Because the distribution of terms in IR collections is very skewed, 5% of the terms occurs in 95% of the documents. Thus, joining the collection tables with a set of query terms with high df constructs large intermediate BATs (`qdj`, `qti`, and `qtfij`), which will degrade performance significantly. Estimating the ranking while leaving out the high df query terms should be much cheaper, since the memory requirements for the intermediate results are much lower, so less swapping will be required.

Although such (domain-specific) optimization techniques have not been used in the current system, it should not be too complicated to integrate them in the mapping from Moa structures to MIL, thanks to the declarative nature of the algebraic approach. A similar argument applies to extending the Mirror DBMS with the buffer management techniques discussed in [JFS98]. In MIL, buffer management is equivalent to directing Monet to load and unload its tables. By integrating such directives in the generated MIL programs, it is expected that these improvements can also be added without many complications.

5.7 QUERY OPTIMIZATION AND MOA EXTENSIONS

This section discusses the opportunities and challenges for query optimization in a multi-model DBMS architecture. It starts with a short example of query optimization at the level of Moa expressions. Next, three types of optimization are identified that affect the mapping from the logical to the physical level. Notice, however, that the current implementation of the Mirror DBMS does not have a query optimizer.

5.7.1 Rewriting logically equivalent expressions

Recall the Moa expression from Example 7, retrieving holiday images that are relevant for the query image. A semantically equivalent expression is given in Example 9. Here, the Moa expression specifies that after computing the beliefs for all images, the only images of interest are those in the 'holiday' category.

Example 9

```
map[THIS.Beliefs](
  select[ THIS.Category = 'holiday' ](
    map[TUPLE< Category:  THIS.Category,
               Beliefs:  sum(getBL(THIS.Content,query,stats))
        > ](
         ( imgs )
)));
```

In most cases, the process of query optimization should rewrite the expression of Example 9 into the logically equivalent expression of 7, before generating the MIL program. The underlying principle in this example is known as the 'push select down' heuristic. Even this straightforward optimization step is hard (if not impossible) to achieve without following an algebraic approach. Conversely, the design of the Mirror DBMS with a strict separation between the logical and physical level can clearly be adapted to perform this style of optimization.

5.7.2 Domain-specific structures can increase efficiency

The content representation in a `CONTREP` structure can also be modelled in Moa without domain-specific extensions, using a nested set of normal tuples; similar to Schek and Pistor's proposal to use NF^2 algebra for the integration of IR and databases [SP82]. Although queries expressing belief computation are much more complicated in a nested relational model, ease of use is not the only motivation for the introduction of the `CONTREP` structure. The domain-specific mapping provided in the implementation of the `CONTREP` structure requires less algebraic operations than the generic mapping of an equivalent query expressed in NF^2. The mapping to MIL of the NF^2 equivalent of Example 5 requires 10 joins and a select; the `CONTREP` version only uses 6 joins, and the same select. Of course, the number of operations is not directly proportional to the amount of processing that is required. But, better efficiency is obtained here, because the designer of domain-specific structures can apply knowledge about the IR process to keep the tables physically synchronized and lexicographically ordered. Notice that this type of performance improvement takes place beforehand, at design time, and does not further improve query processing at runtime.

A similar example relates to the `nidf` operator defined on the `DCSTAT` structure. Without operand, `nidf` returns a tuple, containing a term and its normalized *idf* value, for each term occurring in the indexing vocabulary. Thus, a semijoin between the result of `nidf` without query operand and a collection of query terms is a *logically equivalent* expression for the result of `nidf` with a query as operand (Example 10 shows the queries for both approaches).

Example 10

```
nidf( stats, query );
semijoin[ THIS.nidf, THIS, TUPLE<> ](
  nidf( stats ), query );
```

The DCSTAT structure can compute its answer with fewer operations than the standard Moa query using the semijoin. The domain-specific mapping contributes to the generation of more efficient MIL programs, because it 'knows' in advance the properties of the result.

5.7.3 Defining alternative mappings

In the ideal situation, query optimization searches the complete space of semantically equivalent algebraic expressions for the one expression with the highest estimated performance. Because this search problem is NP-complete, real-life query optimizers only consider a small subspace of all possible candidate expressions. It is more important to avoid very inefficient query plans, than to find the most efficient plan.

Because the designers of Moa structures have detailed knowledge about the high-level process supported by the structure, they could provide a *set* of semantically equivalent algebraic expressions. This set of alternatives may be used as candidate expressions by a query optimizer, reducing the search space significantly. Consider belief computation in the CONTREP structure as an example. When computing a ranking function, two competing strategies can be followed to calculate the required $max\ tf$ values. The first approach computes $max\ tf$ for *all* documents, and then select the values that are needed for the query under consideration:

```
djmaxtfij := {max}( join( dj.mirror, tfij ) )
join( qdj, qdjmaxtfij);
```

The second strategy preselects the document identifiers of documents in which the query terms occur. This way, the normalizing constants $max\ tf$ are only computed for those documents that are relevant for the query.

```
djreq := semijoin( dj.mirror, qdj.mirror );
djmaxtfij := {max}( join( djreq, tfij ) );
join( qdj, qdjmaxtfij);
```

The choice between both evaluation strategies should be made at runtime. The second strategy may be more efficient than the first, if the query terms do not occur in many documents. Although the alternatives in this example may well be generated automatically (it follows the well-known 'push selects down the tree' strategy), more complex logical operators have alternative mappings that are not so likely to be found by a query optimizer at the binary relational level. The rewriting process of Moa structures should therefore be adapted to dynamically rewrite structures, selecting the best alternative based on the run-time data distribution.

5.7.4 Search accelerators in Moa extensions

Finally, a third type of optimization can be identified that should be taken into account when designing structures. Consider once more the example of computing $max\ tf$ values. Instead of computing these results on-the-fly whenever they are needed, this intermediate result can also be stored as a materialized view on BAT `djmaxtfij`. Similarly, access structures may be defined on the results of complex computations.[13]

These types of optimization can of course be hard-coded in the mapping defined in the implementation of structures. But, this violates Codd's notion of index independence; applying search accelerators like materialized views and index structures should be orthogonal to the specification of the mapping of the complex structure to the flattened database model. An open question though is how these accelerators can be defined. In relational databases, the database administrator defines indexes using the SQL `CREATE INDEX` command. But, these indexes are defined on tables that exist in the schema. In this case, the tables and functions for which search accelerators should be defined are hidden within the structure's mapping and may not be visible in the schema at all. The identification of useful search accelerators may only be solved as an additional task for the Moa rewriter, either semi-automatically in interaction with the database administrator, or fully automatically.

5.8 DISCUSSION AND COMPARISON TO OTHER WORK

This section puts the Mirror DBMS in context of other approaches. After discussing its merits, the integration of IR in the Mirror DBMS is compared to approaches extending relational and object-oriented databases. Finally, some other related work is discussed briefly.

5.8.1 Advantages

The combination of Moa and its extensions for IR makes it possible to flexibly combine constraints on the *content* of documents with constraints on their *logical* structure. The nested data model also allows modelling of compound documents. Similarly, several versions of the same underlying collection can be managed easily, e.g. using different stemming algorithms, or extracting concepts with varying NLP techniques. The separation of tasks between the MIL procedures defined in the database and the structural definition of the document collection in Moa supports IR researchers with experimentation. The Moa expressions that define the experiment are invariant to tweaking parameters of the retrieval model.

The clear distinction between the specification of a retrieval model in algebraic operations and its physical execution provides another advantage for experimental research in (multimedia) IR. The researcher can develop new retrieval models without worrying too much about low-level implementation issues like hash tables, index trees and parallelisation. The kernel chooses at runtime from a variety of algorithms to execute the algebraic operations as efficient as possible. Similar to the use of indices in relational databases, we may increase performance by the definition of access structures on the BATs without having to change the code. Furthermore, when executing on

a parallel machine, the IR code automatically benefits from the implementation of parallel kernel algorithms.

These benefits can *in principle* also be achieved using just Monet. However, writing MIL programs on the flattened document representation requires a lot of knowledge of the mapping from documents to BATs. For example, the expression given in Example 7 is translated into a MIL program consisting of 36 operations, and is already quite complex to comprehend, let alone construct. Clearly, Moa expressions have a great advantage over the low-level BAT manipulation. Also, the reduced complexity of the logical Moa expressions makes them easier to manipulate for an algebraic query optimizer. A 'push select down' strategy that first evaluates constraints on the logical structure and then constraints on the content should be easily implemented for Moa expressions; for MIL programs, however, the search space would be too large to accomplish the equivalent rewrite.

5.8.2 Information retrieval and the relational model

The main interest in databases from IR researchers has always been the more traditional database support: concurrent insertion, update, and retrieval of documents, see e.g. [Mac91], [GTZ93], and [DDSS95]. The idea is that, in general, database management systems provide such support, of which the IR systems could benefit. This is certainly regarded an important issue for further study; but, addressing concurrent updates without degrading performance of the complete system is less trivial than it may seem at the first glance. In this thesis, scalability and retrieval have been considered more important problems, that are more or less orthogonal to transaction management.

Early research attempted to represent documents in the data model of a standard DBMS, and express the retrieval model as queries. MacLeod compared the model of text retrieval systems with the data models in relational databases [Mac91], but he concluded that many difficulties arise when mapping a document into a record-oriented structure. Schek and Pistor suggested the use of NF^2 algebra for the integration of IR and databases [SP82]. Written in a time when ADTs were not common, their work is only directly applicable to Boolean retrieval models, which cannot handle the intrinsic uncertainty of the IR process. Also, modelling IR directly in either a relational or a nested-relational DBMS is unlikely to provide sufficient performance to be of any practical usage.

Similar to the development of geographical information systems, researchers then started to build 'coupled' systems: an information retrieval component handles the queries on content and the database component handles the queries on the logical structure. The most primitive approach is to use two completely different systems with a common interface, an example of which is described in [GTZ93]. Modern extensible database systems enable a tighter integration, in which the extension model of the database is used to encapsulate the otherwise completely separated IR component (see e.g. [DDSS95], [VCC96] and [DM97]).

But, the objections against encapsulation given in Chapter 2 apply here equally. Query processing will often fall back to object-at-a-time evaluation instead of the preferred set-orientation. This makes it more complicated to scale up using parallel query processing. The lack of support for inter-operator optimization seems a problem for

multimedia retrieval in particular, because of the larger amount of content abstractions that are necessary. Each feature space requires its own 'miniature' IR component, but query evaluation should avoid the computation of unnecessary intermediate results, for example in feature spaces that will not further influence the final ranking.

5.8.3 IR in an OO-DBMS

Previous approaches to IR and database integration have also suggested the application of object-oriented databases, see [Mil97] for an overview. An object-oriented model is very suited for the implementation of IR. Object data models allow nesting, and in addition to sets, work with multiple collection types. A good example of the design of IR in an OO-DBMS is the Mills's Cobra system, described in [MMR97] and [Mil97].

Mills has developed a very nice environment for rapid development of IR prototypes, that is more easily extended with new IR algorithms compared to the Mirror DBMS. But, its design also has some major drawbacks, most of which are founded in the same objections as have been expressed against OO-DBMSs before. Its design is much more a persistent programming environment than an integration of IR and databases. Like most OO-DBMSs, Cobra is implicitly intended to support some specific application, rather than sharing the same data set across applications. This is very clearly demonstrated in the modelling of collection statistics: Cobra defines the statistics as a property of the document representation *type* rather than a property of some collection. This decision renders it impossible to manage more than one collection in the OO-DBMS, unless these collections share the global statistics. Long-term learning applied to one collection would automatically affect the statistics used for querying the other collection, which is not desirable in all cases.

Representing the content representations in the schema, as proposed in the Mirror DBMS, seems a more elegant solution than making the IR indices explicit like in Cobra. Mills gives the example of one implementation of inverted files that handles updates one at a time, and another that handles them in batches. In the Mirror DBMS, choosing an index structure is orthogonal to defining the schema, and therefore the choice between the two indices is deferred until runtime, and does not have to be known at compile time as in Cobra. Despite of this objection, one of Mills's motivations for adding the indices to the schema is interesting. If the Mirror DBMS would be extended with an inverted file index structure, the content representations would be stored twice: both in the physical representation of the `CONTREP` structure and in the index structure. It may be an interesting idea to extend Monet with 'shadow types', whose instances are only materialized in the index structure. This way, index independence is maintained (as Monet will materialize the values when the index is removed) while the data is only stored once.

Finally, the Cobra system does not help application programmers with obtaining scalability and exploiting parallelism. Instead of increasing data independence, making the index structures explicit in the conceptual schema decreases the data independence. Set-orientation is hard to achieve, because using Cobra involves hard-wiring many nested-loops. Again, although Cobra's object-orientation is very useful for modelling complex objects and their behaviour, the implementation of the objects at the database level should not be the same as the view on the objects at the conceptual level.

Obviously, providing mappings between the different levels of the Mirror DBMS is more complicated than using Cobra's classes, that map directly on the physical level. But, the mappings between different layers in the design and implementation of the Mirror DBMS seem unavoidable if the goal is to develop truly scalable systems.

5.9 SUMMARY

This chapter reported the integration of IR and databases using structural object-orientation, and demonstrated the feasibility of this approach with the prototype implementation of the Mirror DBMS. Various advantages of an integrated approach to IR and databases were identified, and illustrated with the type of queries and data modelling that can be specified in the Mirror DBMS.

The Mirror DBMS prototype has been evaluated for two different applications. First, an advanced image retrieval system has been built to experiment with the multimedia retrieval model, and validate the expressiveness of the language elements provided in Moa's IR extensions. Improvements of the prototype with new algorithms, or a different query formulation process, can be implemented by simply changing the (order of) Moa operators applied. Second, to evaluate if sufficient efficiency can be achieved to be competitive for stand-alone IR systems, the Mirror DBMS has been used to index the TREC collections. Although the efficiency still needs improvement before the current implementation really challenges the more advanced systems, query processing is already fast enough to be a useful platform for IR experiments on TREC's ad-hoc collection.

The belief computations for the retrieval model developed in the previous chapter have been expressed as algebraic operations in the database kernel, with a strong emphasis on set-at-a-time operations. This approach provides desirable data independence between the logical and the physical database. As a result, parallelisation and the selection of access structures have become orthogonal to the development of an information retrieval model. More importantly, this provides the opportunity for query optimization in the mapping from the logical level to the physical level.

Several non-traditional query optimization strategies have been identified, which stimulates further research into the design of the Moa rewriter. First, the domain-specific extensions of Moa allow for a task-specific mapping to operations at the physical level. Second, several semantically identical query translations may be possible for the same Moa expression. A query optimizer at the physical level, that manipulates sequences of operators on binary relations, is likely to have trouble generating these equivalent expressions automatically, due to lack of contextual information. Alternatively, rewriting Moa expressions into MIL query plans should generate several alternative expressions, each defined by the designer of the Moa structure. The best alternative can then be chosen at runtime, using information about the runtime state of the database. Finally, search accelerators like materialized views and indices on path expressions can be used within the mapping from the logical to the physical level.

Notes

1. Note that a complete implementation of the InQuery model also requires proximity operators. Proximity lists should probably be added as atomic types, similar to the polygons in Monet's geographical extensions. This problem has been ignored in this thesis, especially because it is not clear how important the proximity operators are for information retrieval.

2. A `DCSTAT` can also be created manually, using a collection name, the dictionary (or indexing vocabulary), the number of documents in the collection, and the document frequencies (df) of the indexing terms.

3. In later InQuery implementations (since TREC-6 for sure), they have instead used a variant inspired by the Okapi metric.

4. `PLFPICEval` is a more efficient version of the `PICEval` algorithm ($\mathcal{O}(n)$ instead of $\mathcal{O}(n^2)$), for the subclass of `PIC` matrices describing partially linear functions. I actually found a mistake in Greiff's report, and he confirmed that the `PLFPICEval` algorithm had never been implemented in InQuery; which I would consider as (indirect) evidence in favour of the hypothesis that it is more easy to experiment with the implementation of some operators in an algebra for IR than with the code of a complete custom-made IR system.

5. Even when the user enters a short query, it is common to increase the number of query terms significantly using query expansion techniques; a query consisting of more than 100 terms is definitely not outrageous.

6. The difference between these two strategies is quite impressive. For example, the number of possible pairs in the LATIMES collections for TREC topic 311 is 3842250, while the number of really co-occurring terms is only 93241.

7. The texture feature extractors are part of the MeasTex framework, see `http://www.cssip.elec.uq.edu.au/~guy/meastex/meastex.html`.

8. The grid segmentation is performed using PerlMagick, available through CPAN (see URL `http://www.perl.org`).

9. Only positive feedback has been implemented in the current prototype; but, extending the system for handling negative feedback is straightforward.

10. Although more of anecdotal than scientific value, the story of participation in TREC-8 with the Mirror DBMS demonstrates the usability of this architecture for IR experiments. Eight days before the deadline, it still seemed impossible to participate with this year's TREC, as Monet kept crashing while indexing the data; until, the seventh day, the new release suddenly made things work! After a quick consult with Djoerd Hiemstra, I decided to try our luck and see how far we could get. And I admit: it has been a crazy week. It meant running the topics on TREC-6 first, to compare the results with Djoerd's former runs; as well as changing the ranking formula to integrate document length normalization. In the weekend, I made long hours implementing LCA, which turned out to be not so useful as expected. But, in one week I managed to index the data, perform various experiments on the TREC-6 data for calibration, run the best experiments on TREC-8, and submit five runs, just before the final deadline.

11. TREC stands for Text REtrieval Conference.

12. This may seem 'a typical problem of using an academic prototype'. But, Sarawagi et al. report similar memory problems with DB2 when using normal SQL queries for a non-standard application, mining for associations hidden in a large data set [STA98]. Now, a bug in Monet is usually fixed within two or three weeks (see also the acknowledgements!), which I do not suspect to happen with a commercial DBMS used in some non-profit research project. . .

13. In the context of multimedia data, one may even define an access structures like an R-tree on the result of a function, *without* maintaining the results themselves in the database. Consider for example an index on some rarely used feature space. If for some query this feature space seems very appropriate, the access structure is only used as a filter to reduce the set of objects for which the expression then must be recomputed.

6
DATABASES AND DIGITAL LIBRARIES

'If you don't help me, I'll set fire to the woods,' the girl persisted.
That brought the man to his feet, but the girl had already struck one of her matches.
She held it to a tuft of dry grass which flared up instantly.
'God help me!' he shouted. 'The red cock has cowed!'
The girl looked up at him with a playful smile.
'You didn't know I was a communist, did you?'

—Jostein Gaarder, excerpt from *Sophie's world*

6.1 INTRODUCTION

Chapters 3, 4, and 5 focused on the problem of retrieval in multimedia DBMSs. But, other database related functionality can be identified in the design of multimedia digital libraries as well. This chapter argues that multimedia digital libraries are inherently distributed and open. Keeping this in mind, the common approach to constructing a DBMS as a monolithic software system has to be reconsidered. As a solution, this thesis suggests to integrate modern middleware into the DBMS architecture, to ensure an extensible and open framework to connect various participants in the digital library.

The various user groups and their needs motivating an open distributed architecture for multimedia digital libraries, have been studied while spending a summer at the Cambridge Research Laboratory (CRL) (formerly Digital, now Compaq). The original implementation of a digital library prototype for this project, based on Perl scripts, the Postgres extensible relational DBMS, and the HTTP protocol for communication

between its components, has been discussed in [dVEK98]; the project has resulted in some (pending) patent applications as well. Subsequent papers from CRL ([EFI+99] and [Swa99]) have discussed further development of this prototype, among other things by switching to a professional DBMS (Oracle 8) and scaling up to larger amounts of data.

Compared to the CRL project, this chapter presents a more experimental architecture, although it is similar in spirit. It has evolved from the original implementation by a more radical redesign: opening the black box of the relational DBMS, and using CORBA middleware instead of the HTTP protocol. It has been described previously in [dVB98a], [vhHdVBB98], and [vhH98].

The chapter is organized as follows. It starts with the identification of several user groups. The requirements of these users are the basic argument motivating the open distributed architecture proposed in Section 6.3. After discussing the functionality of the architecture's components briefly, an outline of necessary interfaces is given. To test the usability of these ideas, a prototype environment has been implemented. Section 6.5 concludes with some aspects of this design that need more research.

6.2 CHARACTERISTICS OF USER GROUPS

An important observation is that a multimedia digital library involves several more or less independent actors. The following broad classes of actors can be distinguished, based on their role in the digital library [dVEK98]:[1]

- end-users;
- content providers;
- content access providers;
- annotators.

The end-users are the **consumers** of the digital library. They want to search and browse the digital library to retrieve multimedia objects that satisfy their information need. The requirements of this user group has been the major topic of this thesis so far. The other actors are the **producers** in this environment. They collect and manage the multimedia data; their requirements are the topic of this chapter. The first type of producers are the content providers, who own the multimedia data. They decide in which format the data is delivered to the end-users, and they may want to charge money for usage of the footage. Another producer, the content access provider, manages the metadata for searching the collection. Content abstraction (describing the content of multimedia objects using metadata) is the task of the annotators, the third type of producing actors; these include both human annotators and (semi-) automatic metadata extraction software.

A content access provider is not necessarily part of the same enterprise as the content provider. Conversely, a successful access provider may manage metadata about multimedia footage of many different owners. For example, a news archive may provide access to video fragments from both public and commercial broadcasters. And, each owner may desire a different policy to serve the data. The end-user decides

whether to pay for (possibly) better quality data, based on the summary information provided by the content access provider.

Because of copyright issues, content providers are reluctant to have other parties take care of their data. Therefore, it is an unrealistic assumption that the metadata required by the content access provider and the content itself may reside at the same physical location. When multimedia digital libraries start to move out of research laboratories into the real world, legal issues become suddenly a significant factor influencing the success or failure of software that supports the management of a multimedia digital library.

Similar to the different roles for owners and access providers, which can take place in different organizations, the annotation process does not *have to* take place in the same business as the content provider or content access provider. The rise of small innovative enterprises may be expected, developing very specialized extraction software that uses a lot of domain knowledge, e.g. 'face recognition of European politicians'. Such parties will prefer to sell their extraction service rather than their software. Even in the unlikely case when all annotation is performed by the content provider itself, the people involved in (human) annotation typically work in different departments to the ones responsible for content delivery.

6.3 IMPLICATIONS FOR THE DESIGN OF DIGITAL LIBRARIES

The independence of each of these cooperating parties conflicts with the traditional approach of using a centralized database system. Apart from the organizational matters mentioned in the previous section, technical arguments also favour a separation between content and metadata. Searching and browsing with many simultaneous users has different system requirements to streaming multiple video streams; for the latter, see e.g. [Gem95]. Multimedia communication technology develops fast and is best handled by the operating system, as argued in [LMB+96]. Similarly, metadata extraction should be handled separately from its usage in retrieval. The extraction process may involve much computational processing, demand access to special knowledge bases, or use specific hardware. A good example of the latter is the extraction of closed caption from the analog video using special decoder hardware.

6.3.1 An architecture for multimedia digital libraries

Both the requirement of cooperation between independent parties, and the technical considerations mentioned above, make it impossible to use a centralized monolithic DBMS that controls the whole environment. An open distributed architecture is needed, that is in many ways similar to the web. But, instead of leaving the database approach completely behind (a step that advocates of 'intelligent agents' seem to encourage), this thesis argues that management of the digital library requires some central unit of control; which can be achieved with a high-level schema describing the data and operations in the digital library. The database idea of central schema, specified in an abstract data model, is very important for designing multimedia digital libraries. Data independence, separation of the specification and structure of the data from the data itself, is too valuable to disregard completely. Other important benefits of the

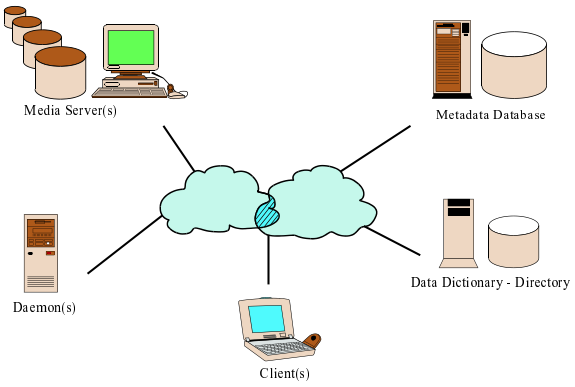


Figure 6.1. An open distributed architecture for multimedia digital libraries.

database approach (such as algebraic query optimization and concurrency control) rely on data independence. Furthermore, as motivated in Chapter 3, content independence is another requirement for multimedia digital libraries, which is impossible to achieve without central coordination as well.

Figure 6.1 shows an open distributed architecture, developed especially for multimedia digital libraries. It offers a solution somewhere in between the two extremes of no control (like the WWW) and complete control (like traditional DBMSs). The architecture consists of clients, daemons, media servers, a metadata database, and the data dictionary/directory[2]. A 'software bus' connects the component systems and provides a single name-space for all components.

The data dictionary/directory manages the specifications of data types and operations on these types. It is also the only interface to access the data, hiding the details of the physical implementation of the data model from the other components. Extensions to and usage of data types or operations must be registered with the data dictionary/directory. It thus has a complete overview of data and operations in the multimedia digital library, providing an opportunity for query optimization, concurrency control and transaction management. This is also the place to implement caching policies and handle security issues. Finally, this overview is used to keep track of necessary contracts between the different participants of the digital library. Content providers will demand guarantees about their property of the data, ensuring that e.g. annotators will not violate the copyright protecting their content. If all access to the data has to pass through the data dictionary/directory, this allows the partners to monitor what happens to the content, and restrict access to multimedia objects that are covered by a legal contract.

The metadata database supports the retrieval functionality, and should be a multimedia DBMS as defined in Chapter 3. Instead of the objects themselves, it stores the locations of multimedia objects (referred to as *object descriptor* in the rest of this chapter) in the media servers, together with all other schema information about the media items, like their relationships to other objects, as well as their content abstractions.

When a new data type is registered in the architecture, the data dictionary/directory creates tables in the metadata database to store (references to) instances of the new type, and it manages the access to these tables. So, the metadata database should be tightly connected to the data dictionary/directory, because the latter drives the query processing in the metadata database. In a research prototype such as the Mirror DBMS, these two components can be fully integrated. In practice, however, people constructing a digital library will prefer a commercial DBMS for storing the metadata. Because a commercial (read relational) DBMS is essentially a black box that cannot be opened, these components will have to be implemented in different software systems, hence the separation in two components in the figure.[3]

In a multimedia digital library, operations include complex metadata extraction techniques, but also relatively simple operations like moving an object from one location to another or performing data format conversions. Because such operations may be implemented by annotators, content providers, as well as the access provider, whose computing equipment may reside at different physical locations in different subnets, the **daemon paradigm** is introduced to provide an extra level of indirection. Daemons perform operations specified in the schema, but they are only loosely connected to the other components. The daemon paradigm is further explained in the following subsection.

Media servers are managed by the content providers. A media server handles the delivery of multimedia data to the clients. It is responsible for playing synchronized audio and video streams. The media servers also communicate with daemons that need the multimedia data for their operations. Because the media servers are separate components, they may be implemented on special hardware with an operating system that is especially suited to handle multiple streams of varying quality of service.

6.3.2 The daemon paradigm

Daemons are components of the architecture performing operations. A daemon requires an input object, and generates an output object; for example, a daemon can be used to transcribe a video object. A daemon can operate on its own initiative (called a client daemon) or on request (called a server daemon). An example of the latter is the implementation of thesauri in the image retrieval prototype discussed in the previous chapter.

Conceptually, a daemon is simply a user-defined function in the multimedia database system. But, unlike user-defined functions in extensible databases, daemon operations are only loosely connected to the metadata database. These operations may be performed on a different machine than the database server, for the purpose of separating the creation of metadata from its management. Thus, this design provides a simple form of (manual) load-balancing; further development of the software bus may enable automatic load-balancing.

Some degree of autonomy, obtained through the notion of client daemons, is considered very important for a multimedia digital library. First, it increases the extensibility of the architecture. In general, the process of (semi-) automatic content abstraction is not well understood. A loosely coupling between the actual code doing the abstraction and the specification of content abstraction techniques in the schema enables

experiments with new algorithms. Also, new products or research prototypes can be integrated without having to produce an ADT for an extensible DBMS, which is usually a complicated task, and certainly impossible without knowing the inner workings of the extraction software. Another (quite pragmatic) argument in favour of loosely coupling is the fact that daemons may crash, especially on slightly different input data than expected, because of different implementations of compression 'standards' and the like. Finally, such autonomy allows each participating organization to move daemons around, perform software upgrades, and assign resources, without depending on some central staff function like a database administrator.

Client daemons, the class of daemons that initiate their operation themselves, request input objects by issuing a `get_work` query. The data dictionary/directory registers the work that has been assigned to each daemon. The daemon has to commit to performing its operation within a certain time; after that time, its input objects may be assigned to a different daemon in subsequent `get_work` calls. At an abstract level, the `get_work` query should specify three conditions: the parent condition p and two child conditions c_1 and c_2. The data dictionary/directory returns a collection of object descriptors for media items that satisfy p, are connected (through relations defined in the schema) to an object fulfilling c_1, but are not connected to an object fulfilling c_2. For example, a daemon may specify that it operates on video objects (p) that have a transcript (c_1) but do not have subtitle annotations (c_2). Most daemons can specify their requested input objects with conditions p, c_1, and c_2 that only use constraints on data types and normal attributes (e.g. 'CNN news bulletins of last week'); these are classified as type-triggered daemons. But, in some cases, it may be important to filter also on the content of the media items. A simple example of such a daemon, classified as content-triggered daemon, is a client daemon that performs speech recognition; the data dictionary/directory must be capable of processing `get_work` queries specifying that the input audio fragments should have a high likelihood to contain speech.

6.3.3 The software bus

The software bus takes care of communication between the various components of the digital library architecture described in the previous sections. Ritter has reviewed different classes of middleware that is used to connect a DBMS with its applications [Rit98]. Quoting Ritter, it is clear that middleware is used in 'normal' database applications for reasons similar to the motivation of the daemon paradigm:

> *If the server is performing an application-specific task for one resource-hungry user, it's unavailable to serve the needs of all other users. It is better to partition this work so hardware resources can focus more accurately on the neediest activities.*

Middleware that supports interoperability between distributed components seems an excellent choice for the realization of Figure 6.1. For example, the CORBA standard (see e.g. [OH97]) supports the low-level functionality required to support daemons in heterogeneous environment. It allows interoperability between daemons developed in different programming languages and running on different operating systems. CORBA's naming service can be exploited for locating currently available daemons, possibly in combination with the property service or the trading service. Once the interfaces of the components have been specified in CORBA's interface definition language

(IDL), any CORBA-compliant object request broker (ORB) provides an implementation of the software bus. In principle, each participant in the digital library may even use a different ORB, in which case CORBA's communication protocol (IIOP) will transparently process requests and replies between the ORBs.

6.4 PROTOTYPE IMPLEMENTATION

This section explains how the proposed architecture can be applied to build a prototype digital library for images collected from the WWW. The software bus is based on the ORBacus (public domain) object request broker.[4] The data dictionary/directory and the metadata DBMS are integrated using the Mirror DBMS.

6.4.1 Interface definitions

To illustrate the interaction between different components of the digital library using CORBA, the interfaces used in the communication with the Mirror DBMS are now described in more detail. The general idea of the interface definitions is that they provide an IDL wrapper around the Mirror DBMS. This makes the conceptual level of the Mirror DBMS (cf. Figure 5.1) available to all participants in the digital library.

Each media item in the digital library has a unique identifier, the object descriptor, which encodes the location of the multimedia object in the media server. Assuming that the media servers and the metadata database share common knowledge about the representation and semantics of these object descriptors, it is sufficient to specify its data type as `any`, which matches the type of any IDL specification.

```
typedef any media_object_descriptor;
```

In the Mirror DBMS, query formulation proceeds by providing examples and relevance judgements. A collection of example objects can simply be represented by a sequence of object descriptors:

```
typedef sequence<media_object_descriptor> id_list;
```

The user's relevance judgement about an object is represented by a structure containing its object descriptor and a long indicating its relevance. A sequence of such structures models the collected relevance judgements in each iteration of the query process.

```
struct query_feedback_item {
  media_object_descriptor id;
  long relevance;
};
typedef sequence<query_feedback_item> query_feedback;
```

For simplicity, it is now assumed that the schema of the metadata database consists of tuples with object descriptors and their associated metadata. The following sample IDL defines data types for the result of queries. The `object` field stores the metadata

for the returned object. The current protocol between client and metadata database does not provide more information about the type of the returned object.

```
struct query_result_item {
  any object;
  media_object_descriptor id;
};
typedef sequence<query_result_item> query_result;
```

Using these definitions, the wrapper around the Mirror DBMS query interface is very simple. This example discusses only retrieval by content; Section 6.5 touches briefly upon the topic of more advanced queries.

```
interface database_wrapper {
  query_result query_by_example(in id_list ids);
  query_result relevance_feedback(in query_feedback f);
}
```

Any client that finds the object request broker of the digital library and uses these interface definitions can now interactively query the Mirror DBMS by example. In his Master's thesis [vhH98], Van het Hof has performed a similar exercise to define the interfaces for the data dictionary/directory, media servers, and client and server daemons. As a result, by implementing the daemon interfaces, any actor in the multimedia digital library can extend the digital library with new algorithms to extract metadata from content.

6.4.2 Demonstration system

A simple web-based image retrieval system demonstrates the use of the wrapped Mirror DBMS in the proposed distributed architecture (see Figure 6.2). In this prototype implementation, the object resolution task of the data dictionary/directory has been delegated to CORBA's naming service. Because the current implementation does not do load-balancing, and assumes that daemons can formulate their `get_work` query directly in Moa, the other tasks of the data dictionary/directory have not been implemented. Despite of these simplifying assumptions, the result has provided sufficient functionality to create the image retrieval prototype discussed in Chapter 5.

To minimize the effort of developing daemons, daemon writers can use an abstract daemon framework. They only have to extend from the base classes provided by this framework, and implement the daemon's core functionality. A 'daemon starter' makes sure the daemon is loaded and initialized correctly, which is based on the builder design pattern (see [GHJV95]).

Six client daemons have been implemented for automatic content abstraction; these implement the two colour and four texture feature extractors mentioned in the previous chapter. These daemons request object descriptors of images that they have not processed before. Two different implementations of server daemons form the 'society of thesauri' in the image retrieval prototype. The first is a wrapper around the Wordnet

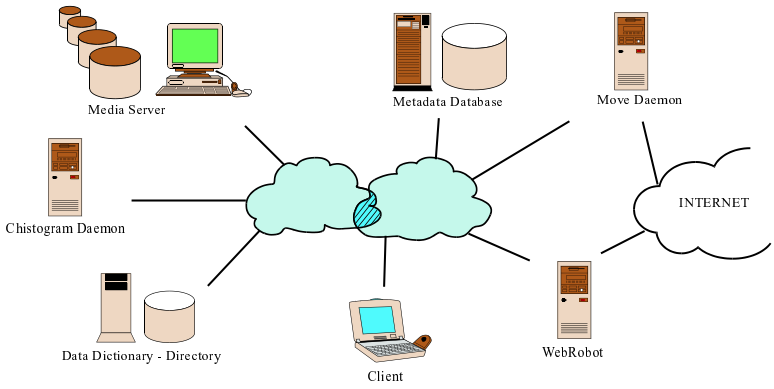


Figure 6.2. The demonstration system.

thesaurus, which can be consulted to expand textual queries. The other implementation of a server daemon uses a private instance of the Mirror DBMS to query association thesauri, automatically constructed from the collection as discussed in Section 5.4.3.

Connected through the software bus, together these components form a digital library. The WebRobot daemon searches the Internet for image URLs, and inserts them into the metadata database. At regular time intervals, the Move daemon submits its `get_work` query, which requests image URLs whose images have not yet been stored on the media server. It then moves a copy of the image to the media server.[5] Notice that in this scenario, the URLs are metadata about the source of the images, and not the object descriptors in the digital library. Next, the feature extraction daemons start to create content abstractions of the image data.[6]

The user interface of the client application consists of two parts: a graphical user interface, and a non-graphical part. The non-graphical part hides the underlying complexity (a façade pattern [GHJV95]) and provides a common high-level interface to GUI designers. When starting a client, the non-graphical part connects to the data dictionary/directory of the digital library, bootstrapping with its IOR (a large number encoding all necessary information to locate an object) retrieved from a publicly known URL (the only parameter determining which digital library will be accessed). Next, the remaining components of the digital library are located through the data dictionary/directory. The current implementation of the client application is a stand-alone Java application.[7]

6.5 DISCUSSION

Every component of the architecture can *in principle* take advantage of the full power of the Mirror DBMS for multimedia retrieval (including content-triggered daemons). But, as mentioned before, in the current implementation the clients have to express their queries directly in Moa, since there exists no proper compiler from OQL to Moa yet. Also, the wrapper hides the schema of the Mirror DBMS behind the interface, which is clearly demonstrated in the definition of `query_result`. To parse the returned

`any` objects, only clients that 'know' the schema can formulate queries against it. This situation should be improved upon with a tight mapping of Moa's data type definitions onto IDL specifications. Furthermore, the definition of `get_work` queries would be greatly improved if it can use a transitive closure operator on the schema. Ideally, the data model and query language would support data modelling with semantic networks, as suggested in [dVEK98].

The complete distributed architecture, including the daemons and the media servers, may together be viewed as a distributed multimedia DBMS. The suggested application of CORBA in this distributed DBMS takes one step beyond using middleware to 'simply' connect between a traditional DBMS and its applications: the middleware is pushed into the implementation of the database management system. This matches the way the CORBA standard develops: it starts to provide more and more services that used to be the domain of DBMSs, such as transaction management and security. But, CORBA itself does not address query optimization and efficient processing of data-intensive operations, as it encourages encapsulation even more than object-oriented database systems; it should not be expected to replace the database approach. More research is required to develop a new type of distributed database management systems that exploit CORBA services maximally, in combination with declarative schema definitions and query processing as is being developed in the So-Simple DBMS and the Mirror DBMS.

6.6 SUMMARY

Examining the environment in which a multimedia digital library typically has to operate, it turns out that the traditional monolithic DBMS architecture is clearly unsuited. An open distributed architecture is the only solution to address copyright issues on the one hand, and widely varying requirements on hardware and software on the other hand. This chapter has proposed to develop a new type of distributed DBMS, in which advanced middleware is an integrated part of its architecture. A first prototype implementation has been presented, based on the Mirror DBMS and CORBA middleware. This implementation has been applied effectively to build an image library that is populated with images retrieved by a Web robot. The integration of CORBA and database approach looks promising. But, much research remains to be done, especially with respect to load-balancing, updates, recovery, and security.

Notes

1. This categorization is largely based on informal talks between researchers of the Cambridge Research Laboratory and media archivers of WGBH, a part of the US Public Broadcasting System; see also [dVEK98, EFI+99, Swa99].

2. This is called the data dictionary/directory to distinguish it from the (local) data dictionary of the metadata database system.

3. The reluctance of database vendors to start development of DBMSs with a different architecture than those developed in the 1970s explains the success of so-called *application servers*. Application servers provide the type of functionality attributed to the data dictionary/directory in the distributed architecture described in this chapter. Such software manages extra information about the data and its applications, outside the scope of the DBMS. For administrative applications, such extra information is usually referred to as the 'business logic'.

4. ORBacus is available from URL `http://www.ooc.com`.

5. In the prototype implementation of the architecture, the media server is just a simple Java server that uses the file system to store its data.

6. Unfortunately, not all steps of Figure 5.4 have been implemented as daemons yet. This is mainly due to the fact that a CORBA language binding for Perl is still very experimental, and both image segmentation and clustering of the feature vectors use Perl and its modules extensively.

7. A preliminary version of the client application could run within the Netscape web browser, whose private ORB communicated transparently with the ORB of the multimedia database. Unfortunately, varying bugs in different versions of the browser's implementation (for different platforms) render this not a very usable solution yet.

7
THE EVALUATION PROBLEM

karma police
arrest this man
he talks in maths
he buzzesLikeAfridge
hes like a detuned radio.

—Radiohead, *Karma police*

7.1 INTRODUCTION

Today, there are no satisfactory methods for measuring the effectiveness of multimedia search techniques. Precision and recall types of metrics have been used in some of the literature but are impractical due to the tedious process of measuring relevances. The process is complicated because of human subjectivity in tasks involving multimedia. Also, multimedia collections quickly grow very large, making evaluation expensive with respect to the required hardware. There seem to be no standard corpora or benchmark procedures.

In the previous chapters, the functionality provided in the Mirror DBMS has been evaluated by building actual systems, and its performance has been tested on the TREC collection. Measuring its effectiveness for multimedia retrieval has been discussed only shortly. The main purpose of this chapter is to explain the problems with evaluation of multimedia information retrieval, and demonstrate that most previous 'evaluations' reported in literature are neither very useful, nor very convincing. Another goal is to

Table 7.1. Overview of amount of papers reviewed

topic	# papers	# projects
audio/speech	7	4
image features	14	14
image databases	20	9
video	7	6
web, multimedia	8	4

identify the basic ingredients of a better evaluation procedure; for, proper evaluation is a necessary step to further improve multimedia retrieval techniques.

This chapter starts with a review of evaluation approaches that have been taken in literature about multimedia retrieval systems. Table 7.1 summarizes the distribution of the studied papers over relevant research topics. Papers dealing with image retrieval have been divided in two categories, one focused on feature representations themselves and one on the retrieval of images using these representations. The web/multimedia category consists of papers on research projects that attempt to address retrieval of multimedia documents rather than base objects of some particular new medium. The last field provides a rough estimate of the number of different research projects that are reported upon in the papers used. The review covers far from all related publications, but the sample of papers studied is sufficiently large to draw some sad conclusions, which are the topic of Section 7.5. Section 7.6 identifies some important aspects that should be addressed in the development of a better evaluation methodology for multimedia retrieval systems.

A summary of the literature review presented in this chapter has been published before as part of [PMS+98].

7.2 QUANTATIVE EVALUATION

Quantitative evaluation uses corpora, for which a set of queries with previously collected relevance judgments provide a ground-truth for evaluation. Although mainly used in information retrieval community, with a variety of test collections available for public use, other research fields have started to develop similar collections.

7.2.1 IR evaluation methodology

Information retrieval research has developed a strong scientific evaluation methodology. Using large data collections, and ground-truth data for a set of queries, the quality of the retrieval systems is measured and then used to compare different approaches. Evaluation with ground-truth data in traditional IR is based on the following assumptions:

- The user reads all the results;
- Reading one relevant document does not influence the judgment of other documents in the result set;

- Relevance is considered a binary property.

Under these assumptions, the quality of the result for a query can be expressed using recall (how many of the relevant documents in the collection have been retrieved?) and precision (how many of the retrieved documents are relevant?).

For systems that produce ranked output, it is not likely that the user checks *all* query results. This is resolved by choosing several cut-off points, to represent the fact that not every user will evaluate as many documents of the result set, and computing average recall and precision values for these sets. A common method is to compute the **11-point average precision**. This measure is computed by averaging the precision over the standard recall points (0%, 10%, 20%, etc.). To get the precision for these standard recall points, precision and recall are calculated for each relevant document in the result set and interpolated.

Comparison of the performance of different approaches requires a decision whether the observed difference in performance is statistically significant. For, IR experiments are stochastic experiments affected by random errors. It is not sufficient to compare different retrieval approaches by their mean performances, because these can be heavily affected by outliers. Instead, the distributions of the observations should be compared. The statistical significance of the performance difference is best checked using non-parametric tests, because these tests make least assumptions on the experimental data. The sign-test and the paired t-test (which is parametric though) are most widely used in IR.

7.2.2 Test collections

Text Many different collections are available to evaluate retrieval performance of full-text retrieval systems. The Cranfield collection is an old collection that is rather small, but has complete relevance judgements. The TREC collection is more popular because of its more realistic size, but it has incomplete relevance judgements due to the pooling technique, as discussed in Section 5.6. The complete data set consists of approximately five Gigabytes of text documents with relevance judgments for 400 topics. The data set is continuously expanding, especially with documents in different languages for the evaluation of cross-lingual retrieval systems, and with a static subset of the WWW for the evaluation of web retrieval. Another benefit of the TREC collection is that it comes with some scripts to process the output of retrieval systems to compute the interpolated precision at various recall points, and compare the performance between different systems.

Speech Schäuble and Wechsler did innovative work in speech retrieval with an open vocabulary [SW95, Sch97]. For this work, they developed a test collection of 1289 documents with relevance judgments for 26 queries [WS95].[1] A different collection was constructed by Cambridge University together with Olivetti research, to evaluate their video mail retrieval system. For evaluation purposes, they collected relevance assessments for 50 requests on a (fairly small) collection of 300 messages, the VMR1 message set [JFJY96, BFJ+96].

The situation with test collections for speech retrieval has started to improve, since in TREC-6, the Spoken Document Retrieval (SDR) track has been started, aimed at the

evaluation of speech retrieval systems. The collection consists of a human transcribed text and the output of a speech recognizer of the same data. The TREC-6 collection consisted of 50 hours of broadcast news, and 49 queries with one relevant document per query [VH97]. The TREC-8 SDR corpus is significantly larger. It consists of approximately 385 hours of broadcast news, in about 900 files, for which 50 topics have been defined. Because the same data set has been used for TREC-7 SDR, 23 topics have relevance judgements. Evaluation is done in two conditions, with or without known story boundaries.

Other audio Musclefish is a small company in Silicon Valley that specializes in audio retrieval techniques. They developed feature extraction for audio retrieval, and evaluated the performance against a manually classified collection [WBKW96]. The collection is rather small; it consists of about 400 samples varying from 1 to 15 seconds, that has been manually classified in groups like 'bells', 'crowds', and 'laughter'. For his evaluation of machine learning techniques for audio retrieval, generalized from speaker identification research, Foote [Foo97] too used the collection gathered by Musclefish, but only provides details about classifying the data into a small amount of groups. In a demonstration on the web, Foote applied the same classifier to search music based on similarity, but this has not been evaluated.

Image The Brodatz texture collection is used often to evaluate image retrieval algorithms; e.g., in this review, [SC94], [LP96], [MM96], [RSZ96], [ZCA97], [PP97], [MP97], [CSZSM97], [SJ], and [VL99] based their evaluation on the Bordatz collection. The complete Brodatz database consists of 112 classes of each 9 images [PMS96]. But, most of these papers mention only the existence of 112 images, and a small subset of 13 homogeneous textures is really used widely. In some of these papers, the Brodatz set is used to construct larger data sets, using images composed of several pieces of different textures.

VisTex, provided by the MIT Media Lab, has 167 textures from natural scenes, categorized into 19 mutually exclusive groups. This collection is used in [Min96] and [ZCA97]. The MIT papers also use a collection of 96 vacation photo's, e.g. [MP97] and [PM95], and another data set called the 'BT collection' (also used in Chapter 5) [Min96]. De Bonet (of the MIT AI Lab) used a collection based on 29 Corel CD, each containing 100 images of some thematic class [Bon97, BV97]. Later, he evaluated his texture features on a SAR data set for vehicle classification in radar images [BVI98]. Note that he argues that the Brodatz set is 'too easy' for proper evaluation of texture models [BV98]. But, he does not mention that the images in the Corel collection are not very suited for the evaluation of image retrieval systems either. For, the number of images per class is high in comparison to the number of classes, making the retrieval task of similar images (defined as images from the same class) too easy.

The MeasTex initiative of University of Queensland provides a common framework for evaluation of texture classification algorithms.[2] This framework contains software and test suites necessary to measure performance of an algorithm, as well as implementations of and results for some well known texture classification algorithms. The MeasTex framework rates an algorithm based on its average performance

on a test-suite. The data set consists of the Brodatz collection and the VisTex collection, supplemented with the MeasTex images (artificial and natural textures) and the 'Ohanian and Dubes' images. Although the task of texture classification is only partially related to building multimedia retrieval systems, the availability of a common evaluation framework for this subtask is definitely a step in the right direction.

Gevers and Smeulders evaluated the effectiveness of several color models for color invariant retrieval [GS96]. They use a collection of 500 images of 2-D and 3-D domestic objects, recorded with a CCD color camera. They randomly selected a set of 70 objects, and recorded those objects again (so these have different lighting conditions) to create a query set. De Bonet did some similar experiments with the Corel data, by varying visual characteristics of some of the images (such as brightness, contrast, and varying degrees of additional noise) and measuring the effect on retrieval [BV97, Bon97].

Some initiatives aim to create huge image collections using the WWW as a source. Sclaroff et al. developed a fully operational system ImageRover [STC97]. Their fleet of 32 robots is estimated to collect approximately 1 million images monthly. To ensure diversity of images, they start the robot at several Yahoo categories. The Dutch national SION-AMIS project[3] has started to construct the large Acoi benchmark [NK98]. The benchmark is geared at provision of 1 million still images, hundreds of video sequences, and thousands of audio tracks. Unfortunately, the current setting of these benchmarks only measure the execution performance, not the quality of the retrieved images, rendering the data set useless for evaluation of effectiveness.

Video None of the reviewed papers has performed an evaluation experiment to measure the quality of video retrieval. But, some papers reviewed the qualitity of scene segmentation algorithms. Gargi and Kasturi [GK96] evaluated different color spaces and frame difference measures used in video segmentation algorithms. They constructed a test set of 21,000 frames from 9 movies, with 200 ground truth cuts. The human subjects first previewed the video data at full-speed, and then marked the cuts during half-speed viewing. They found that this procedure resulted in the most consistent results. Zhang et al. [ZKS93] used only three videos.

The MITRE corporation used an adapted version of the Text-tiling algorithm (see [Hea94]) on the captions of broadcast news to find program boundaries [MHMG97]. The adapted algorithm did not only find topic boundaries, but provided topic classifications. They collected captions for 17 1/2 hours of video data, but did not hand-segment topic boundaries, only program boundaries. They did notice that the program changes that they missed using the text had visual cues, so an algorithm combining text and vision might work better.

7.3 REDUCING THE COST OF EVALUATION

The most significant problem with the creation of test collections for multimedia retrieval is the cost involved; both in time (of researchers and test users), and, especially in case of high-quality video data, in the required hardware for storage and distribution. Another problem is the definition of the ground truth: which multimedia objects are

relevant for some task? This section discusses some approaches that somehow reduce these problems, without limiting the applicability of the results too much.

Lakshmi Ratan and Grimson evaluated the performance of their method for the classification of scenes in images without a manually annotated image collection. They compared the retrieval performance of their method to the performance of QBIC [NBE+93] on the same data set, counting the number of false positives in the result sets of both systems [RG97]. The collection used is created from the Corel photo library, this particular version consisting of 800 annotated images of natural scenes.

La Cascia et al. also avoid manual annotation of the data set [CSS98]. They describe a small user study to measure the performance of their retrieval system collecting images from the web ([STC97]). In an evaluation, they randomly select 100 images from a 10,000 image collection. The subjects then have the task to try and find those images using the retrieval system. A search is considered successful if the subjects could get the image displayed in the top 100 within four iterations of relevance feedback. They vary the following conditions: use visual, textual, or both for content representation, and use visual, textual, or both for the relevance feedback. The experiments have been repeated for larger database sizes, up to 100,000 images. Only two subjects have been studied however, and these were the same in the different conditions.

A similar experimental setup has been used by Papathomas et al. [PCC+98], there described as the 'identical-target testing paradigm'. This paper is unique in the thoroughness of experimental evaluation, and is further discussed in the beginning of Section 7.6.

Minka uses identical-target testing as well, but evaluated the algorithms in his Foureyes learning agent on learning time using simulation instead of real users [Min96, PMS96]. He measured the number of corrections that a user would have to supply until the database system 'knows' the correct classification for all objects, taking this number as an approximation of the quality of the system.

Lienhart et al. describe the MoCA video abstracting project in [LPE97]. They avoid the problem of generating ground truth for the evaluation, using commercial video abstracts broadcast on German television, to compare with the quality of the generated abstracts. The evaluation experiment had a setup similar to the Turing-test. They found that the human subjects could not tell the difference between the automatically generated abstracts and the commercial abstracts. Notice however that they did not evaluate whether it is possible to do *worse*, by using some randomly selected fragments of the video data as a baseline performance.

Vasconcelos and Lippman presented a Bayesian video modeling framework for segmentation and classification of the segments [VL97]. They performed some experiments to select the best movie classification algorithm, using 24 trailers of two minutes duration. This small collection contains about 100 shots *(which consumes 26 Gb disk space though)*. They used the manual classification from the 'Internet Movie Database' to evaluate the quality of the automatic classification against.

7.4 MINIMAL EVALUATION

Another approach to reduce the cost of evaluation is to perform only some minimal evaluation as 'proof-of-concept'. A great deal of published multimedia retrieval research barely has an evaluation phase. The techniques are explained, and the results of a small set of example queries are given to convince us that the techniques work. Although some authors seem to realize the relative weaknesses in these evaluations, others are perfectly happy with the 'proof' derived from their experiments. The following list gives some examples of multimedia retrieval projects, that claim to have evaluated their systems:

- The QBIC image retrieval project by IBM was very influential, which demonstrated the potential usage of image querying by similarity for the first time [NBE+93]. In [FBF+94], the effectiveness of the color-based image retrieval has been measured for a relatively small database (1000 images for 10 queries). Retrieval by shape has been evaluated on an even smaller scale (259 test images and 7 queries).

- The Chabot project at Berkeley [OS95] used the Postgres DBMS to implement image retrieval techniques. They evaluated the system using a single query ('yellow flower'), on a database containing 11,643 images (with 22 yellow flowers).

- Vellaikal and Kuo (UCLA) used a set of 3,400 color images with four queries in the ground-truth [VK95]. In [VK96] 12,000 images were used in an evaluation experiment with three queries.

- The MARS project at University of Illinois ([MRC+97]) used a collection from the Getty museum to experiment with shape based retrieval. This collection has 300 images of ancient African artifacts . In [ORC+97], the quality of retrieval has been evaluated for thirteen conceptual queries, like 'stone masks' or 'golden pots'. In other work, they did not evaluate the quality of the retrieval, but they did study the relevance information provided by users from the system [RHMO97a, RHMO97b, RHM98]. Their main conclusion is that different users have very different measures of what images are 'similar', and hence need relevance feedback methods.

- The work on multimedia retrieval from the National University of Singapore. They developed search engines for retrieval of faces (FACEit) and trademarks (STAR) [NL95]. STAR has been evaluated against a database of 500 trademarks, in which two 'ideal' ranked retrieval sets have been constructed by ten people (using voting to get agreement). The results of the system are compared to this ideal result [WLM+97]. The same group developed the more general content-based retrieval engine CORE [WNM+95]. For the evaluation of content-based retrieval of segmented images, Chua et al. used a small collection of about 100 images divided in 10 different categories [CLP94].

- Columbia University, NY, has projects on both image and video retrieval. Smith and Chang evaluate color retrieval in their VisualSEEK system using a collection 3100 images using a single request to select the 83 images of lions in the collection [SC96, SC97]. The video retrieval system VideoQ, allows some dynamic aspects

in the query language [CCM+97]. They use a collection of 200 shots, categorized into sports, science, nature, and history. For evaluation they used only 4 queries.

- In the Informedia project at Carnegie Mellon University, an impressive amount of tools for video retrieval have been implemented [WKSS96]. But, for evaluation of their News-on-demand application, only the quality of the output of the speech recognizer has been evaluated [HW97].

- Ardizzone and La Cascia describe the JACOB video retrieval system developed at University of Palermo [AC97]. Their evaluation uses a collection of 1500 keyframes extracted from about 500 shots. They use a test set of five example queries, for topics 'water polo', 'interview', 'TV-show', 'TV-movie', and 'cycling race'.

- The research of Sheikholeslami et al. of State University of New York at Buffalo includes the usage of clustering techniques to improve retrieval. In [SCZ98a] they evaluated their system with 29,400 texture and color feature vectors (without mentioning the number of images used). The database is classified in five classes (cloud, floral, leaves, mountain, water). 'Recall' and 'precision' ratios are used to express the quality of the resulting clusters. Retrieval is evaluated using 19 query images. In [SCZ98b] they evaluate a neural network to learn weights (off-line) for combining feature spaces during retrieval. In this paper, they use judgments for 400 pairs of images, classified as similar or non-similar. They neither mention how many different images are used to construct the set of pairs, nor who did classification.

7.5 DISCUSSION

Summarizing, we may conclude that there exists no appropriate methodology for the evaluation of a multimedia retrieval system. The traditional information retrieval methodology using test collections with ground-truth is hard to apply to the multimedia case for two reasons. First, increasing the size and widening the scope of collections faces high costs. Most collections are small, or consist of a small amount of rather homogeneous groups (like the collections based on the Corel data). Some collections have been tailored to evaluate only a very specific low-level task, e.g. the evaluation of texture algorithms based on the Brodatz collection. Larger collections can be gathered from the WWW with relatively low cost, but consequently have no relevance judgements attached. Second, in most application scenario's in which content-based retrieval techniques seem useful, there exists no objective ground truth, because this 'truth' will depend heavily on the user. Therefore, it seems unlikely that an evaluation methodology without input from several real users can draw valid conclusions about the effectiveness of some multimedia retrieval system.

Together, these two observations can be summarized as **the evaluation problem** of multimedia retrieval. As a result of this problem, the effectiveness of multimedia retrieval systems (including the image retrieval prototype based on the Mirror DBMS) has never been evaluated well. This lack of evaluation makes it hard to say anything useful about the relative value of the different retrieval systems that have appeared in

literature. Although this conclusion may seem to disqualify the field as unscientific, that is not the intended message of this chapter: multimedia retrieval research is still in its infancy, and the introduction of new techniques is therefore viewed as more important than thorough (but expensive) experimental evaluation of prototypes that are not yet powerful enough for real applications. But, the lack of evaluation methodology will become more and more a limiting factor in the development of the field of multimedia retrieval, and this problem should be placed high on the research agenda.

Until a better approach is found to measure the performance of multimedia retrieval systems, it is very important that researchers realize the limitations of their experimental 'proof'. Far too often, the results on some small collection with a narrow scope are generalized to very different, high-level search tasks. Most papers claim to present a novel, *better* approach to multimedia query processing. As proof that the novel approach really is 'better', results provided in the paper's evaluation section are vaguely based on concepts borrowed from the scientific evaluation methodology used in IR. The evaluation sections in the papers mentioned in Section 7.4 proof however mainly the authors' lack of understanding of that methodology, as:

- only a small number of queries is used (often one or two);
- only precision-recall measures for one cut-off point in a ranking are presented;
- a significance test is not applied;
- the data is divided in a small number of classes, and these are considered both the relevance judgments and the complete description of the user's information need;
- relevance judgments are considered completely objective, but usually made by the paper's authors and not by real users.

Also, multimedia retrieval constitutes typically much more to end-users than 'just' the identification of objects of some particular class. The inherent subjectivity of multimedia search is usually ignored completely. Almost all evaluation experiments with multimedia search test only the success on a task of object identification; e.g. does the image contain a lion or not.[4] The emotional and aesthetic values, that play an important role in the evaluation process of the user, are overlooked. Or, even worse, the underlying techniques are 'improved' in such a way that they are less sensitive to exactly those aspects that *are* important for such values.

7.6 TOWARD BETTER TEST COLLECTIONS

This section makes some recommendations for future evaluation experiments measuring effectivity of retrieval in multimedia digital libraries. The first comments are directly related to the construction and usage of test collections:

- Big isn't always better;
- Define the baseline performance.

First, the variety in content of the images in the collection may matter more than the size of a data set. Second, a baseline performance is needed to compare with.

Both issues are illustrated perfectly by [PCC+98], which reports experiments with the PicHunter system. These authors found that even a random presentation of images performs 'surprisingly well when users are given the ill-defined task of judging when images are similar rather than identical to a query'. Another quote from this paper: 'This casts doubt on the strength of results given in other reports where no baseline is established'. The possible explanation they offer is based on the nature of the Corel collection that they used. Because of its small number of relatively large thematic clusters, the probability of selecting randomly an image from the same cluster as the query image is quite high. Remarkably, they found also that a version of PicHunter that *only* uses textual annotation, and no other content-based features, performed best with finding the target image identical to the query image. Although this can also be explained by the nature of the collection, it provides also another indication that content-based metadata are still not adequately modelling human perception, despite of all the 'great' results presented in the reviewed papers.

The inherent subjectivity in multimedia searching makes it impossible to develop a test suite that is not based on a user task that is performed by real users. Because one user task can be quite different from another, proper evaluation of a multimedia retrieval model requires the evaluation with different test sets. These different sets should evaluate different aspects of the retrieval model of the system to be evaluated, and in particular:

- Does it effectively support the user's task?
- Can it adapt to the user's subjectivity?

In general purpose photo archives, e.g. in the context of journalism, it seems possible to improve existing (text-based) search systems with content-based retrieval techniques, to assist the user with browsing large intermediate result sets acquired with broad text queries. Sormunen et al. [SMJ99] have further developed the rough outline of [PMS+98] into a proposal of a task-oriented evaluation framework and an efficient procedure for constructing test collections for this task. The aim of the framework is to base the evaluation of the retrieval system on the photo similarity judged by users in the context of a real task: in their case, journalists performing an illustration task. The browsable sets of relevant photos are retrieved using textual search. The idea is to let the journalists define what perceptual similarity is between all relevant photos (the *user-perceived similarity*), and later use these judgements for evaluation of content-based retrieval systems. The advantage of the evaluation framework is that the normal (text-based) systems in the photo archives can be used to construct the test set, independent of the algorithms in the system to be evaluated. The recall base is built on the test user's similarity judgements, so these are subjective. Thus, the performance of the content-based retrieval algorithms can be evaluated on realistic criteria.

With image collections, problems with object identification may easily hide the problems with adapting to the subjective aspects of the user's query. If the user searches a 'sad image of a lion', an image of a small kitten with a sad atmosphere is probably of less value than a 'happy' image of a lion. The music domain provides a context that is better suited to evaluate how the query process adapts to subjectivity of the users. Especially in music, aesthetic value is more important than objective notions

like words, melody, or instruments used. It seems therefore a good idea to investigate the use of a music retrieval task for measuring a retrieval model's quality with respect to adapting to subjectivity. A drawback is that content abstraction of music is not easy, and the success criteria are vaguely defined and implicit to the user.

Finally, test collections for multimedia retrieval grow automatically when people use them, because the interactive style of multimedia querying creates new relevance judgements. When image retrieval systems become available at the WWW, or are used in a real workplace, it is important to collect this 'free' input about their operation:

- Log the interaction between user and system;
- Give users the opportunity to enter comments.

Historical data of experiments with real users, on a publicly known data set, may proof a very valuable source of information. It allows to replay the interaction, and analyze reported 'mistakes' of the system. Given a collection and a user task, such data is crucial to allow comparison in the laboratory between different approaches. Of course, such data should become available just like the collection itself, in a format suitable for exchange, so that the development of multimedia retrieval systems can benefit from the same type of interaction between different research groups as is common in the text retrieval community.

7.7 SUMMARY

This chapter has revealed 'the evaluation problem' of multimedia retrieval, which is caused by the cost of creating test collections on the one hand, and the high degree of subjectivity inherent to multimedia retrieval on the other hand. It reviewed the evaluation approaches of a reasonable amount of scientific literature about multimedia retrieval systems. Common mistakes have been identified, made when borrowing from the quantitative IR evaluation methodology without fully understanding its underlying assumptions. The chapter concluded with some reflections on important aspects of evaluating multimedia retrieval systems. Further research into multimedia retrieval systems can only be successful if the evaluation problem is addressed more seriously, and the development of new prototypes is based on real applications, with real users, and real user needs.

Notes

1. Their original proposal of a speech retrieval system, published in [GS92], provides a good example of the risks involved with laboratory experiments. By using a text collection, they attempted to evaluate their proposed retrieval system without building a speech recognizer. But, the simulated speech processing assumed the spaces between words in the text corpus were known: they overlooked the fact that a real speech recognizer would never be capable of detecting word boundaries.

2. See URL `http://www.cssip.elec.uq.edu.au/~guy/meastex/meastex.html`.

3. More information is available at URL `http://www.cwi.nl/~acoi/Amis/`.

4. The (Australian) researcher David McSquire, who works on the Viper system, characterized this type of experiment (in personal communication) as the 'quest for the kangaroo detector'.

8
CONCLUSIONS

The struggle itself towards the heights
is enough to fill a man's heart.
One must imagine Sisyphus happy.

—Albert Camus, *The Myth of Sisyphus*

8.1 SUMMARY

A database management system is a general-purpose software system that facilitates the processes of defining, constructing, and manipulating databases for various applications. The main characteristic of the 'database approach' is that it increases the value of data by its emphasis on data independence. DBMSs, and in particular those based on the relational data model, have been very successful at the management of administrative data in the business domain.

This thesis has investigated data management in multimedia digital libraries, and its implications on the design of database management systems. The main problem of multimedia data management is providing access to the stored objects. The content structure of administrative data is easily represented in alphanumeric values. Thus, database technology has primarily focused on handling the objects' logical structure. In the case of multimedia data, representation of content is far from trivial though, and not supported by current database management systems.

The information retrieval (IR) community has since long studied the retrieval of text documents by their content. Also, research in topics like computer vision and image

analysis has led to content-based retrieval techniques for querying image and audio collections. Retrieval systems based on these ideas are typically standalone systems that have been developed for very specific applications. There is not much consensus on the integration of these techniques in general-purpose DBMSs. State-of-the-art solutions simply make new functions available in the query language. These functions interface to otherwise still standalone software systems. This leaves to the user the burdens of both query formulation and the combination of results for each single representation into a final judgement. Also, this leads to inefficient query processing for queries involving several content representations.

Like any DBMS, a MM-DBMS is a general-purpose software system that supports various applications; but, the support is targeted to applications in the specific domain of digital libraries. Four new requirements have been identified for this domain: (1) multimedia objects can be active objects, (2) querying is an interaction process, (3) query processing uses multiple representations, and (4) query formulation provides content independence. The Mirror architecture and its implementation in the Mirror DBMS therefore provide basic functionality for the management of both the content structure and the logical structure of multimedia objects.

In the Mirror DBMS, content management and databases are completely integrated. Recognizing the strong relationship with IR query processing, the inference network retrieval model has been adapted for multimedia retrieval. The logical algebra of the DBMS has been extended with operators for probabilistic inference in this retrieval model. This approach to integration enables the study of new query optimization techniques, and simplifies the introduction of parallellism and distribution in IR query evaluation.

Other characteristics of multimedia digital libraries demand the support for distribution of both data and operations, and extensibility of data types and operations. As a solution, the integration of advanced middleware and database technology is proposed to replace the monolithic design of traditional database systems. Again, this idea has been worked out in a prototype implementation.

The proposed MM-DBMS architecture has been evaluated in three ways, using the Mirror DBMS prototype implementation. First, the advantages of the integration of content management in the Mirror DBMS have been illustrated by several example queries capturing different information needs. The multimedia IR model developed in this thesis has been tested in some small-scale experiments in the domains of music and image retrieval, confirming that reasoning with multiple representations is both possible and useful. Finally, the execution performance of IR query processing has been evaluated using a standard text retrieval benchmark.

8.2 CONCLUSIONS

Recall the research questions motivating this dissertation:

- Can we identify requirements with respect to data management that are specific for applications in a multimedia digital library?

- If so, can we support these requirements in a subclass of DBMSs (that will be called *multimedia DBMSs*); that is, without violating the design principles (especially the

notion of data independence) that characterize 'the database approach' to data management?

- If so, can we provide this support in an efficient and scalable manner?

Regarding the first research question, Chapter 3 has clearly shown that multimedia data management requires specific facilities, that are not supported in traditional database systems. A 'normal' DBMS stores metadata about the *structure* of the data in its catalog. But, the query formulation problem of (multimedia) information retrieval requires multimedia applications to reason about the content of the stored data. Therefore, multimedia DBMSs have to manage metadata about the *content* of the data as well. Content independence has been identified as a desirable property with respect to the management of such content-based metadata.

With respect to the second research question, the Mirror architecture discussed in Chapter 4 addresses the new requirements through the identification of three components: the data abstraction component, the content abstraction component, and the retrieval engine. Strict separation between the latter two enforces content independence. It has been argued that the retrieval engine should preferably be based on the Bayesian formalism, by adapting probabilistic retrieval models that have been developed previously in IR. Chapter 5 has explained how the retrieval engine can be integrated with a DBMS, by using the multi-model DBMS architecture proposed in Chapter 2. Contrary to other approaches, this maintains data independence, and does not obstruct the goal of efficient set-oriented processing. Furthermore, developing a new retrieval model has become equivalent to changing the queries that express its characteristics, in a high-level, declarative database language.

The third research question has not been answered completely. Although the design of the Mirror DBMS has aimed for scalability and efficiency, the current prototype is not sufficiently efficient to challenge the best stand-alone retrieval systems. But, it seems possible to overcome these efficiency problems by improving the mapping between the logical and physical levels of the architecture.

Another contribution of this thesis may have been less explicit, being slightly hidden beneath the more technical discussions. But, this dissertation also demonstrated the importance of considering the end-user in the design of complex software systems. It emphasizes that the foundation of a software architecture for multimedia digital libraries should be based on the interaction pattern between user and system. A digital library can only become useful if it supports a dialogue in which both cooperate to achieve the goal of satisfying the user's information need. For, in multimedia retrieval it is most important to find out what the user wants. This observation establishes a relationship between the design of multimedia retrieval systems and cognitive science.

8.3 FURTHER WORK

The current result of this research is that the overall structure of a MM-DBMS has been determined. It is now clear how the interaction between database and IR model can take place through Moa. The core functionality works, and has been tested in some small case studies. But, a lot of work needs to be done. The current system does not have an implementation at the conceptual level, only at the logical and physical levels.

More development is required for a user interface at the conceptual level, and also for a smooth integration of the open distributed architecture with the current implementation of the metadata database. As mentioned several times, the efficiency of the mapping between the logical and physical levels should be improved as well. At the moment, the specification of Moa queries is done by hand, which should be generated from a high-level language like OQL with extensions for multimedia. A tool is needed to translate the end-user schemas without content-based metadata into internal database schemas, and to derive an appropriate inference network from this schema.

The experimental evaluation of this work has been rather limited. Unfortunately, evaluation of the effectiveness of multimedia retrieval systems has been shown problematic. The development of an evaluation methodology for multimedia retrieval systems is probably the most important topic for further research. Once an acceptable evaluation framework has been set up, Bayesian learning techniques should be studied for both short-term and long-term learning. A more advanced dialogue between user and system may be especially beneficial to improve multimedia information retrieval. Also, retrieval models that integrate structure and content in a common reasoning framework are of interest.

Finally, this thesis has mainly looked at retrieval problems. But, many other topics are significant challenges for data management in digital libraries. Transaction processing and security have not been addressed in this thesis, but are very important issues. Maybe, the middleware in the open distributed architecture presented in Chapter 6 can help in addressing these problems. Furthermore, the desire for distribution of operations and data is not unique in multimedia applications; it may apply equally in the business domain. Further research should investigate whether opening the black box of relational database systems, and unifying their functionality with the functionality provided in so-called application servers, will lead to better performing systems, as well as increased data independence.

References

[AB87] M.P. Atkinson and O.P. Buneman. Types and persistence in database programming languages. *ACM Computing Surveys*, 19(2), June 1987.

[ABC+76] M.M. Astrahan, M.W. Blasgen, D.D. Chamberlin, K.P. Eswaran, J.N. Gray, P.P. Griffiths, W.F. King, R.A. Lorie, P.R. McJones, J.W. Mehl, G.R. Putzolu, I.L. Traiger, B.W. Wade, and V. Watson. System R: relational approach to database management. *ACM Transactions On Database Systems*, 1(2):97–137, 1976.

[AC97] E. Ardizzone and M. La Cascia. Automatic video database indexing and retrieval. *Multimedia tools and applications*, 1(4):29–55, 1997.

[ACD+97] M. Asgarian, M.J. Carey, D.J. DeWitt, J. Gehrke, J.F. Naughton, and D.N. Shah. The BUCKY object-relational benchmark. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 135–146, Tucson, Arizona, May 1997.

[Atk89] M.P. Atkinson. The object-oriented database system manifesto. In *Proceedings of the first international conference on deductive and object-oriented databases*, Kyoto, Japan, 1989.

[AY96] N. Adam and Y. Yesha. Strategic directions in electronic commerce and digital libraries: towards a digital agora. *ACM Computing Surveys*, 28(4):818–835, December 1996.

[Bar95] J.D. Barrow. *The artful universe*. Little, Brown and company, 1995.

[BCF97] E. Bertino, B. Catania, and E. Ferrari. *Multimedia Databases in Perspective*, chapter Query Processing, pages 181–217. Springer Verlag, 1997.

[BFJ+96] M.G. Brown, J.T. Foote, G.J.F. Jones, K. Spärck Jones, and S. J. Young. Open-vocabulary speech indexing for voice and video mail retrieval. In *Proceedings ACM Multimedia*, Boston, November 1996.

[Bis95] C.M. Bishop. *Neural networks for pattern recognition*. Oxford university press, Oxford, 1995.

[BK95] P.A. Boncz and M.L. Kersten. Monet: An impressionist sketch of an advanced database system. In *BIWIT'95: Basque international workshop on information technology*, July 1995.

[BK99] P.A. Boncz and M.L. Kersten. MIL primitives for querying a fragmented world. *The VLDB Journal*, 1999. To appear.

[BKS98] S. Boll, W. Klas, and A. Sheth. *Multimedia data management. Using metadata to integrate and apply digital media*, chapter Overview on using metadata to manage multimedia data, pages 1–24. In Sheth and Klas [SK98], 1998.

[BMK99] P.A. Boncz, S. Manegold, and M.L. Kersten. Database architecture optimized for the new bottleneck: Memory access. In *Proceedings of 25th International Conference on Very Large Databases (VLDB '99)*, Edinburgh, Scotland, UK, September 1999. To appear.

[Bon97] J.S. De Bonet. Novel statistical multiresolution techniques for image synthesis, discrimination, and recognition. Master's thesis, MIT, 1997.

[BQK96] P.A. Boncz, C.W. Quak, and M.L. Kersten. Monet and its geographic extensions. In *Proceedings of the 1996 EDBT conference*, 1996.

[Bro95] E.W. Brown. *Execution performance issues in full-text information retrieval*. PhD thesis, University of Massachusetts, Amherst, October 1995. Also appears as technical report 95-81.

[Bun96] W. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on knowledge and data engineering*, 8(2):195–210, April 1996.

[BV97] J.S. De Bonet and P. Viola. Structure driven image database retrieval. In *Advances in Neural Information Processing*, number 10, 1997.

[BV98] J.S. De Bonet and P. Viola. Texture recognition using a non-parametric multiscale statistical model. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 1998.

[BVI98] J.S. De Bonet, P. Viola, and J.W. Fisher III. Flexible histograms: A multiresolution target discrimination model. In E.G. Zelnio, editor, *Proceedings of SPIE*, volume 3370, 1998.

[BWK98] P.A. Boncz, A.N. Wilschut, and M.L. Kersten. Flattening an object algebra to provide performance. In *Fourteenth International Conference on Data Engineering*, pages 568–577, Orlando, Florida, February 1998.

[Cal94] J.P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland, July 1994.

[Cam96] R. Camps. Domains, relations and religious wars. *SIGMOD Record*, 25(3), September 1996.

[Cat91] R.G.G. Cattell. What are next-generation database systems? *Communications of the ACM*, 34(10):31–33, October 1991.

[CB97] R.G.G. Cattell and D.K. Barry, editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann Publishers Inc., 1997.

[CC85] R. Chellappa and S. Chatterjee. Classification of textures using Gaussian Markov random fields. *IEEE Transactions on Acoustics Speech and Signal Processing*, 33:959–963, 1985.

[CCM+97] S.-F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong. VideoQ: an automated content based video search system using visual cues. In *Proceedings of ACM Multimedia 1997*, Seattle, November 1997.

[CD96] M. Carey and D.J. DeWitt. Of objects and databases: a decade of turmoil. In Vijayaraman et al. [VBMS96], pages 3–14.

[CH80] R.W. Conners and C.A. Harlow. A theoretical comparison of texture algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:204–222, 1980.

[CK85] G.P. Copeland and S.N. Koshafian. A decomposition storage model. In *Proceedings of the SIGMOD Conference*, pages 268–279, 1985.

[CLP94] T.-S. Chua, S.-K. Lim, and H.-K. Pung. Content-based retrieval of segmented images. In *ACM Multimedia 94*, pages 211–218, San Francisco, 1994.

[Clu98] S. Cluet. Designing OQL: allowing objects to be queried. *Information systems*, 23(5):279–305, 1998.

[CLvRC98] F. Crestani, M. Lalmas, C.J. van Rijsbergen, and I. Campbell. "Is this document relevant? . . . probably": a survey of probabilistic models in information retrieval. *ACM Computing Surveys*, 30(4):528–552, December 1998.

[CMMY98] I.J. Cox, M.L. Miller, T.P. Minka, and P.N. Yianilos. An optimized interaction strategy for Bayesian relevance feedback. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, 1998.

[Cod70] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[Col94] M. Colton. Illustra: The multi-media DBMS. Technical report, Illustra Information Technologies, Inc., 1994.

[Coo90] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[CP85] S. Ceri and G. Pelagatti. *Distributed databases. Principles and systems*. McGraw-Hill, 1985.

[CS95] P. Cheeseman and J. Stutz. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1995.

[CSK93] B.B. Chaudhuri, N. Sarkar, and P. Kundu. Improved fractal geometry based texture segmentation technique. *IEEE Proceedings*, 140:233–241, 1993.

[CSS98] M. La Cascia, S. Sethi, and S. Sclaroff. Combining textual and visual cues for content-based image retrieval on the World Wide Web. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, Santa Barbara, CA, June 1998.

[CSZSM97] W. Chang, G. Sheikholeslami, A. Zhang, and T. Syeda-Mahmood. Efficient resource selection in distributed visual information systems. In *ACM Multimedia'97*, Seattle, WA, November 1997.

[CvR95] F. Crestani and C.J. van Rijsbergen. Probability kinematics in information retrieval. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'95)*, pages 291–299, 1995.

[Dat85] C.J. Date. *An introduction to database systems*. Addison-Wesley, third edition, 1985.

[DD98] C.J. Date and H. Darwen. *Foundation for Object/Relational Databases: the third manifesto*. Addison-wesley, 1998.

[DDSS95] S. DeFazio, A. Daoud, L.A. Smith, and J. Srinivasan. Integrating IR and RDBMS using cooperative indexing. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'95)*, pages 84–92, 1995.

[Dit88] K.R. Dittrich. *Advances in object-oriented database systems*, volume 334 of *Lecture notes in computer science*, chapter Preface. Springer-Verlag, 1988.

[Dit91] K.R. Dittrich, editor. *On object-oriented database systems*, chapter Object-oriented database systems: the notion and the issues, pages 3–10. Springer-Verlag, 1991.

[DM97] S. Dessloch and N. Mattos. Integrating SQL databases with content-specific search engines. In *Proceedings of the 23rd VLDB conference*, Athens, Greece, 1997.

[dV98] A.P. de Vries. Mirror: Multimedia query processing in extensible databases. In *Proceedings of the fourteenth Twente workshop on language technology (TWLT14): Language Technology in Multimedia Information Retrieval*, pages 37–48, Enschede, The Netherlands, December 1998.

[dVB98a] A.P. de Vries and H.M. Blanken. Database technology and the management of multimedia data in Mirror. In *Multimedia Storage and Archiving Systems III*, volume 3527 of *Proceedings of SPIE*, pages 443–455, Boston MA, November 1998.

[dVB98b] A.P. de Vries and H.M. Blanken. The relationship between IR and multimedia databases. In *The 20th IRSG colloquium: discovering new worlds of IR*, Grenoble, France, March 1998.

[dVEK98] A.P. de Vries, B. Eberman, and D.E. Kovalcin. The design and implementation of an infrastructure for multimedia digital libraries. In *Proceedings of the 1998 International Database Engineering & Applications Symposium*, pages 103–110, Cardiff, UK, July 1998.

[dVvDBA99] A.P. de Vries, M.G.L.M. van Doorn, H.M. Blanken, and P.M.G. Apers. The Mirror MMDBMS architecture. In *Proceedings of 25th International Conference on Very Large Databases (VLDB '99)*, Edinburgh, Scotland, UK, September 1999. Technical demo.

[dVvdVB98] A.P. de Vries, G.C. van der Veer, and H.M. Blanken. Let's talk about it: Dialogues with multimedia databases. Database support for human activity. *Displays*, 18(4):215–220, 1998.

[dVW99] A.P. de Vries and A.N. Wilschut. On the integration of IR and databases. In *Database issues in multimedia; short paper proceedings, international conference on database semantics (DS-8)*, Rotorua, New Zealand, January 1999.

[Edw93] B. Edwards. *Drawing on the right side of the brain*. Harper Collins Publishers, London, 1993.

[EFI+99] B. Eberman, B. Fidler, R.A. Ianucci, C. Joerg, L. Kontothanassis, D.E. Kovalcin, P. Moreno, M.J. Swain, and J.-M. Van Thong. Indexing multimedia for the internet. Technical report, Cambridge Research Laboratory, March 1999. Also appears at Visual '99.

[EK95] M.W. Eysenck and M.T. Keane. *Cognitive Psychology. A student's handbook*, chapter Mental representation. Lawrence Erlbaum Associates, 3 edition, 1995.

[EN94] R. Elmasri and S.B. Navathe. *Fundamentals of database systems*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, second edition, 1994.

[EV94] B. Eaglestone and R. Vertegaal. Intuitive human interfaces for an audio-database. In *Proceedings of the Second International Workshop on Interfaces to Database Systems (IDS94)*. Lancaster University, 1994.

[Fal96] C. Faloutsos. *Searching multimedia databases by content*. Kluwer Academic Publishers, Boston/Dordrecht/London, 1996.

[FBF+94] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3:231–262, 1994.

[FF95] R.M. Fung and B.A. Del Favero. Applying Bayesian networks to information retrieval. *Communications of the ACM*, 38(3):43–48, March 1995.

[FLGD87] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30:964–971, November 1987.

[Foo97] J. Foote. A similarity measure for automatic audio classification. In *Proc. AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, March 1997.

[FS89] I. Fogel and D. Sagi. Gabor filters as texture discriminator. *Journal of Biological Cybernetics*, 61:103–113, 1989.

[GCT98] W. Greiff, W.B. Croft, and H. Turtle. PIC matrices: A computationally tractable class of probabilistic query operators. Technical Report IR-132, The Center for Intelligent Information Retrieval, 1998. submitted to ACM TOIS.

[Gem95] D.J. Gemmell. Multimedia storage servers: A tutorial. *IEEE Computer*, 28(5):40–49, May 1995.

[GFJ98] W.I. Grosky, F. Fotouhi, and Z. Jiang. *Multimedia data management. Using metadata to integrate and apply digital media*, chapter Using metadata for the intelligent browsing of structured media objects, pages 123–148. In Sheth and Klas [SK98], 1998.

[GHJV95] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[GHK+96] Nita Goyal, Charles Hoch, Ravi Krishnamurthy, Brian Meckler, and Michael Suckow. Is gui programming a database research problem? In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 517–528. ACM Press, 1996.

[GJ97] A. Gupta and R. Jain. Visual information retrieval. *Communications of the ACM*, 40(5):70–79, May 1997.

[GK96] U. Gargi and R. Kasturi. An evaluation of color histogram based methods in video indexing. In *Proceedings of the first international workshop on Image databases and multi-media search (IDB-MMS '96)*, pages 75–82, Amsterdam, The Netherlands, August 1996.

[Gra93] G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2):73–170, June 1993.

[GS92] U. Glavitsch and P. Schäuble. A system for retrieving speech documents. In *Proceedings of the 15th annual international SIGIR*, pages 168–176, Denmark, June 1992.

[GS96] T. Gevers and A.W.M. Smeulders. A comparative study of several color models for color image invariant retrieval. In *Proceedings of the first international workshop on Image databases and multi-media search (IDB-MMS '96)*, pages 17–26, Amsterdam, The Netherlands, August 1996.

[GTZ93] J. Gu, U. Thiel, and J. Zhao. Efficient retrieval of complex objects: Query processing in a hybrid DB and IR system. In *Proceedings of the 1st German National Conference on Information Retrieval*, 1993.

[Gut84] A. Guttman. R-trees: a dynamical index structure for spatial searching. In *Proceedings of the SIGMOD Conference*, pages 47–57, Boston, June 1984.

[Hal99] J.Y. Halpern. A counterexample to theorems of Cox and Fine. *Journal of Artificial Intelligence Research*, 10:67–85, 1999.

[Har98] H.L. Hardman. *Modelling and Authoring Hypermedia Documents*. PhD thesis, University of Amsterdam, 1998.

[HBH88] E.J. Horvitz, J.S. Breese, and M. Henrion. Decision theory in expert systems and artificial intelligence. *International journal of approximate reasoning*, 2:247–308, 1988. (Special issue on Uncertainty in Artificial Intelligence).

[HC93] D. Haines and W.B. Croft. Relevance feedback and inference networks. In *Proceedings of the sixteenth annual international ACM SIGIR conference on research and development in information retrieval (SIGIR'93)*, pages 2–11, 1993.

[HCL+90] L.M. Haas, W. Chang, G.M. Lohman, J. McPherson, P.F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. Carey, and E. Shekita. Starburst mid-flight: As the dust clears. *IEEE Trans. on Knowledge and Data Engineering*, 2(1):143–160, March 1990.

[Hea94] M.A. Hearst. Multi-paragraph segmentation of expository text. In *ACL '94*, Las Cruces, 1994.

[Hec95] D. Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Advanced technology division, March 1995. Revised edition November 1996.

[Hie98] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In Ch. Nicolaou and C. Stephanidis, editors, *Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries: ECDL '98*, pages 569–584. Springer Verlag, 1998.

[HJ98] A. Hampapur and R. Jain. *Multimedia data management. Using metadata to integrate and apply digital media*, chapter Video data management systems: metadata and architecture, pages 245–286. In Sheth and Klas [SK98], 1998.

[HM95] R.S. Heller and C.D. Martin. A media taxonomy. *IEEE MultiMedia*, 2(4):36–45, Winter 1995.

[Hor99] K.S. Van Horn. If it ain't Bayesian, it's broken: an examination of Cox's theorem. Appeared on UAI mailing list, UAI@cs.orst.edu, June 1999.

[HRT+94] M.-Y. Hwang, R. Rosenfield, E. Thayer, R. Mosur, L. Chase, R. Weide, X. Huang, and F. Alleva. Improving speech recognition performance via phone-dependent VQ codebooks and adaptive language models in SPHINX-II. In *Proceedings of ICASSP-94*, pages 549–552, 1994.

[HS95] A. Hauptmann and M. Smith. Text, speech, and vision for video segmentation: The informedia project. In *AAAI Fall 1995 Symposium on Computational Models for Integrating Language and Vision*, 1995.

[HT98] Laura M. Haas and Ashutosh Tiwary, editors. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. ACM Press, 1998.

[HW97] A.G. Hauptmann and M.J. Witbrock. *Intelligent multimedia information retrieval*, chapter Informedia: news-on-demand multimedia information acquisition and retrieval, pages 215–239. AAAI Press/MIT Press, 1997.

[HWC95] A.G. Hauptmann, M.J. Witbrock, and M.G. Christel. News-on-demand - an application of informedia technology. *D-LIB Magazine*, September 1995.

[Iac93] J. F. Iaccino. *Left brain-right brain differences*. Lawrence Erlbaum Associates, Publishers, 1993.

[Jay90] E.T. Jaynes. *Maximum-Entropy and Bayesian Methods*, chapter Probability theory as logic, pages 1–16. Kluwer, Dordrecht, 1990.

[Jay91] E.T. Jaynes. *Maximum entropy and Bayesian methods*, chapter Notes On Present Status And Future Prospects, pages 1–13. Kluwer, Dordrecht, 1991.

[Jay96] E.T. Jaynes. Probability theory: The logic of science. The book will be published by Cambridge university press. Available online from http://bayes.wustl.edu/etj/prob.html., 1996.

[JC94] Y. Jing and W.B. Croft. An association thesaurus for information retrieval. Technical Report 94-17, University of Massachusetts, 1994.

[JFJY96] G.J.F. Jones, J.T. Foote, K. Spärck Jones, and S.J. Young. Retrieving spoken documents by combining multiple index sources. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, Zürich, Switzerland, August 1996.

[JFS98] B.Th. Jónsson, M.J. Franklin, and D. Srivastava. Interaction of query evaluation and buffer management for information retrieval. In Haas and Tiwary [HT98], pages 118–129.

[JK84] M. Jarke and J. Koch. Query optimization in database systems. *ACM Computing Surveys*, 16(2):111–152, June 1984.

[KdVB97] Wolfgang Klas, Arjen de Vries, and Christian Breiteneder. *Multimedia Databases in Perspective*, chapter Current and emerging applications, pages 13–30. Springer Verlag, 1997.

[Ken79] W. Kent. Limitations of record-based information models. *ACM Transactions On Database Systems*, 4(1):107–131, March 1979.

[Kic96] G. Kiczalis. Beyond the black box: open implementation. *IEEE Software*, 8-11, 1996.

[Kim94] W. Kim. Observations on the ODMG-93 proposal for an object-oriented database language. *SIGMOD Record*, 23(1), March 1994.

[KN98] M.L. Kersten and N.J. Nes. Fitness joins in the ballroom. CWI, July 1998.

[Kra98] P.J. Krause. Learning probabilistic networks. *Knowledge Engineering Review*, 13(4), 1998. To appear.

[Lap02] P.S. Laplace. *A philosophical essay on probabilities*. Wiley and Sons, New York, 1902. Translated from the 6th French edition, 1812.

[LLOW91] C.W. Lamb, G. Landis, J.A. Orenstein, and D.L. Weinreb. The ObjectStore system. *Communications of the ACM*, 34(10):50–63, October 1991.

[LLPS91] G.M. Lohman, B. Lindsay, H. Pirahesh, and K.B. Schiefer. Extensions to starburst: objects, types, functions, and rules. *Communications of the ACM*, 34(10):79–92, October 1991.

[LMB+96] Ian Leslie, Derek McAuley, Richard Black, Timothy Roscoe, Paul Barham, David Evers, Robin Fairbairns, and Eoin Hyden. The Design and Implementation of an Operating System to Support Distributed Multimedia Applications. *IEEE Journal on Selected Areas in Communication*, 14(7):1280–1297, September 1996.

[LP96] F. Liu and R.W. Picard. Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):722–733, July 1996.

[LPE97] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video abstracting. *Communications of the ACM*, 40(12), December 1997.

[Mac91] I.A. Macleod. Text retrieval and the relational model. *Journal of the American society for information science*, 42(3):155–165, 1991.

[MHMG97] I. Mani, D. House, M. Maybury, and M. Green. *Intelligent multimedia information retrieval*, chapter Towards content-based browsing of broadcast news video, pages 241–258. AAAI Press/MIT Press, 1997.

[Mil97] T.J. Mills. *Content modelling in multimedia information retrieval systems. The Cobra retrieval system*. PhD thesis, University of Cambridge, July 1997.

[Min96] T.P. Minka. An image database browser that learns from user interaction. Master's thesis, MIT, 1996. Also appeared as MIT Media Laboratory technical report 365.

[Mit97] T.M. Mitchell. *Machine learning*. McGraw-Hill, 1997.

[Miz98] S. Mizzaro. How many relevances in information retrieval? *Interacting With Computers*, 10(3):305–322, 1998. In press.

[MJL76] G.A. Miller and P.N. Johnson-Laird. *Language and perception*. Cambridge university press, 1976.

[MM96] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.

[MMR97] T.J. Mills, K. Moody, and K. Rodden. Cobra: a new approach to IR system design. In *Proceedings of RIAO'97*, pages 425–449, July 1997.

[MP97] T.P. Minka and R.W. Picard. Interactive learning using a "society of models". *Pattern Recognition*, 30(4), 1997.

[MRC+97] S. Mehrotra, Y. Rui, K. Chakrabarti, M. Ortega, and Th.S. Huang. Multimedia analysis and retrieval system. In *Proceedings of the 3rd International Workshop on Information Retrieval Systems*, Como, Italy, September 1997.

[MRT91] C. Meghini, F. Rabitti, and C. Thanos. Conceptual modeling of multimedia documents. *IEEE Computer*, 24(10):23–30, October 1991.

[MS98] M. Markkula and E. Sormunen. Searching for photos - journalists' practices in pictorial IR. In *The challenge of image retrieval*, Newcastle upon Tyne, UK, 1998. University of Northumbria.

[NBE+93] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, and C. Faloutsos. The QBIC project: querying images by content using color, texture and shape. Technical Report RJ 9203, IBM Research Division, 1993.

[NK98] N. Nes and M. Kersten. The Acoi algebra: A query algebra for image retrieval systems. In *Advances in Databases. 16th British National Conference on Databases, BNCOD 16*, pages 77–88, Cardiff, Wales, UK, July 1998.

[NKJ96] N.J. Nes, M.L. Kersten, and A. Jonk. Database support for line clustering. In *ASCI conference*, pages 277–282, Vosse-Meren, June 1996.

[NL95] A. Desai Narasimhalu and M.-K. Leong. Experiences with content based retrieval of multimedia information. In *Final Workshop on Multimedia Information Retrieval (MIRO'95)*, October 1995.

[OH97] R. Orfali and D. Harkey. *Client/Server programming with JAVA and CORBA*. John Wiley & Sons, Inc., 1997.

[Oor98] H. Oortwijn. Content-based retrieval van muziek. Master's thesis, University of Twente, Database group, August 1998. In Dutch.

[ORC+97] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, and Th.S. Huang. Supporting similarity queries in MARS. In *Proceedings of ACM Multimedia 1997*, Seattle, Washington, November 1997.

[OS95] V.E. Ogle and M. Stonebraker. Chabot: retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, September 1995.

[Pap95] C. Papadimitriou. Database metatheory: asking the big queries. In *PODS*, pages 1–10, 1995.

[Par96] S. Parsons. Current approaches to handling imperfect information in data and knowledge bases. *IEEE Transactions on knowledge and data engineering*, 8(3):353–372, June 1996.

[PCC+98] Thomas V. Papathomas, Tiffany E. Conway, Ingemar J. Cox, Joumana Ghosn, Matt L. Miller, Thomas P. Minka, , and Peter N. Yianilos. Psychophysical studies of the performance of an image database retrieval system. In *Proc. SPIE*, 1998.

[Pea88] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of Plausible Inference*. Morgan Kaufmann, California, 1988.

[PM95] R.W. Picard and T.P. Minka. Vision texture for annotation. *Journal of multimedia systems*, 3:3–14, 1995.

[PMS96] R.W. Picard, T.P. Minka, and M. Szummer. Modeling user subjectivity in image libraries. In *IEEE International Conference on Image Processing*, Lausanne, September 1996.

[PMS+98] A. Mark Pejtersen, M. Markkula, E. Sormunen, M. Tico, and A.P. de Vries. Evaluation method for content-based photo retrieval. In *Mira Workshop*, Dublin, October 1998.

[PP97] K. Popat and R.W. Picard. Cluster-based probability model and its application to image and texture processing. *IEEE Transactions on Image Processing*, 6(2):268–284, February 1997.

[RG97] A. Lakshmi Ratan and W.E.L. Grimson. Training templates for scene classification using a few examples. In *Proceedings IEEE Workshop on Content-based Access of Image and Video Libraries*, San Juan, Puerto Rico, June 1997.

[RHM98] Y. Rui, Th.S. Huang, and S. Mehrotra. Relevance feedback techniques in interactive content-based image retrieval. In *Proceedings of IS&T and SPIE Storage and Retrieval of Image and Video Databases VI*, San Jose, CA, January 1998.

[RHMO97a] Y. Rui, Th.S. Huang, S. Mehrotra, and M. Ortega. Automatic matching tool selection via relevance feedback in MARS. In *Proc. of The 2nd Int. Conf. on Visual Information Systems*, San Diego, California, December 1997.

[RHMO97b] Y. Rui, Th.S. Huang, S. Mehrotra, and M. Ortega. A relevance feedback architecture for content-based multimedia information retrieval systems. In *Proceedings IEEE Workshop on Content-based Access of Image and Video Libraries*, San Juan, Puerto Rico, June 1997.

[Rit98] D. Ritter. The middleware muddle. *SIGMOD Record*, 27(4), December 1998.

[RM96] B.A.N. Ribeiro and R. Muntz. A belief network model for IR. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 253–260, Zürich, Switzerland, August 1996.

[Row95] N. C. Rowe. Retrieving captioned pictures using statistical correlations and a theory of caption-picture co-reference. In *Fourth Annual Symposium on Document Analysis and Retrieval*, Las Vegas, April 1995.

[RSZ96] E. Remias, G. Sheikholeslami, and A. Zhang. Block-oriented image decomposition and retrieval in image database systems. In *The 1996 International Workshop on Multi-media Database Management Systems*, Blue Mountain Lake, New York, August 1996.

[SB91] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1), 1991.

[SC94] J.R. Smith and S.-F. Chang. Quad-tree segmentation for texture-based image query. In *ACM Multimedia 94*, pages 279–286, San Francisco, 1994.

[SC96] J.R. Smith and S.-F. Chang. VisualSEEk: a fully automated content-based image query system. In *ACM Multimedia 96*, Boston, MA, 1996.

[SC97] J.R. Smith and S.-F. Chang. *Intelligent multimedia information retrieval*, chapter Querying by color regions using the VisualSEEK content-based visual query system, pages 23–41. AAAI Press/MIT Press, 1997.

[Sch97] P. Schäuble. *Multimedia information retrieval. Content-based information retrieval from large text and audio databases*. Kluwer Academic Publishers, 1997.

[SCZ98a] G. Sheikholeslami, W. Chang, and A. Zhang. Semquery: Semantic clustering and querying on heterogeneous features for visual data. In *ACM Multimedia 98*, Bristol, UK, September 1998.

[SCZ98b] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Neuromerge: An approach for merging heterogeneous features in content-based image retrieval systems. In *4th International Workshop on Multi-Media Database Management Systems (IW-MMDBMS'98)*, Dayton, Ohio, August 5-7 1998.

[Sen75] M.E. Senko. Information systems: records, relations, sets, entities, and things. *Information systems*, 1(1):3–13, 1975.

[Ses98a] P. Seshadri. Enhanced abstract data types in object-relational databases. *The VLDB Journal*, 7(3):130–140, 1998.

[Ses98b] P. Seshadri. Predator: A resource for database research. *SIGMOD Record*, 27(1), March 1998.

[SJ] S. Santini and R. Jain. Similarity matching. Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

[SK91] M. Stonebraker and G. Kemnitz. The POSTGRES next-generation database management system. *Communications of the ACM*, 34(10):79–92, October 1991.

[SK98] A. Sheth and W. Klas, editors. *Multimedia data management. Using metadata to integrate and apply digital media*. McGraw-Hill, 1998.

[SLR96] P. Seshadri, M. Livny, and R. Ramakrishnan. The design and implementation of a sequence database system. In Vijayaraman et al. [VBMS96], pages 99–110.

[SM95] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *CHI'95 Proceedings*, Denver, CO, USA, 1995.

[SM99] M. Stonebraker and Dorothy Moore. *Object-relational DBMSs: Tracking the next great wave*. Morgan Kaufmann Publishers, Inc., second edition, 1999.

[SMJ99] E. Sormunen, M. Markkula, and K. Järvalin. The perceived similarity of photos - seeking a solid basis for the evaluation of content-based retrieval algorithms. In *Final Mira Conference*, Glasgow, April 14-16 1999. To appear in British Computer Society (BCS) *electronic* Workshops in computing.

[SMMR99] D. McG. Squire, W. Müller, H. Müller, and J. Raki. Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback. In *The 11th Scandinavian Conference on Image Analysis*, pages 7–11, Kangerlussuaq, Greenland, June 1999.

[SP82] H.-J. Schek and P. Pistor. Data structures for an integrated data base management and information retrieval system. In *Proceedings of the Eighth International Conference on Very Large Data Bases*, pages 197–207, Mexico City, 1982.

[SRH90] M. Stonebraker, L.A. Rowe, and M. Hirohama. The implementation of POSTGRES. *IEEE Transactions on Knowledge and Data Engineering*, 2(1):125–142, March 1990.

[SS86] H.-J. Schek and M.H. Scholl. The relational model with relation-valued attributes. *Information systems*, 11(2):137–147, 1986.

[STA98] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating mining with relational database systems: Alternatives and implications. In Haas and Tiwary [HT98], pages 343–354.

[STC97] S. Sclaroff, L. Taycher, and M. La Cascia. Imagerover: A content-based image browser for the world wide web. In *Proceedings IEEE Workshop on Content-based Access of Image and Video Libraries*, San Juan, Puerto Rico, June 1997.

[Ste95] H. Steenhagen. *Optimization of object query languages*. PhD thesis, University of Twente, The Netherlands, 1995.

[Sub98] V.S. Subrahmanian. *Principles of multimedia database systems*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.

[SW95] P. Schäuble and M. Wechsler. First experiences with a system for content based retrieval of information from speech recordings. In *IJCAI Workshop: Intelligent multimedia information retrieval*, August 1995.

[Swa99] M.J. Swain. Searching for multimedia on the world wide web. Technical report, Cambridge Research Laboratory, March 1999. Invited paper at ICMCS '99.

[SWKH76] M. Stonebraker, E. Wong, P. Kreps, and G. Held. The design and implementation of INGRES. *ACM Transactions On Database Systems*, 1(3):189–222, 1976.

[SZ96] A. Silberschatz and Stan Zdonik. Strategic directions in database systems - breaking out of the box. *ACM Computing Surveys*, 28(4):764–778, December 1996.

[TC91] H. Turtle and W.B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions of information systems*, 9(3), 1991.

[TC92] H.R. Turtle and W.B. Croft. A comparison of text retrieval models. *The computer journal*, 35(3):279–290, 1992.

[TK78] D. Tsichritzis and A. Klug. The ANSI/X3/SPARC DBMS framework report of the study group on database management systems. *Information systems*, 3:173–191, 1978.

[Tur91] H.R. Turtle. *Inference networks for document retrieval*. PhD thesis, Univeristy of Massachusetts, 1991.

[VBMS96] T.M. Vijayaraman, A.P. Buchmann, C. Mohan, and N.L. Sarda, editors. *Proceedings of the 22nd VLDB conference*, Mumbai (Bombay), India, September 1996.

[VCC96] S.R. Vasanthakumar, J.P. Callan, and W.B. Croft. Integrating INQUERY with an RDBMS to support text retrieval. *Bulletin of the technical committee on data engineering*, 19(1):24–34, March 1996.

[vD99] M.G.L.M. van Doorn. Thesauri and the mirror retrieval model. Master's thesis, University of Twente, Database group, July 1999.

[Vel98] D.D. Velthausz. *Cost-effective network-based multimedia information retrieval*. PhD thesis, University of Twente, 1998.

[VH97] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Sixth Text Retrieval Conference (TREC-6)*, number 500-240 in NIST Special publications, November 1997.

[vhH98] E. van het Hof. Een architectuur voor multimedia databases. Master's thesis, University of Twente, Database group, 1998. In Dutch.

[vhHdVBB98] E. van het Hof, A.P. de Vries, H.E. Blok, and H.M. Blanken. Een architectuur voor multimedia databases. In *CIW '98: Bouwen aan het informatielandschap*, Antwerp, December 1998. In Dutch.

[VK95] A. Vellaikal and C.-C. J. Kuo. Content-based image retrieval using multiresolution histogram representation. In *SPIE Digital Image Storage and Archiving Systems*, pages 312–323, Philadelphia, October 1995.

[VK96] A. Vellaikal and C.-C. J. Kuo. Joint spatial-spectral indexing for image retrieval. In *IEEE International Conference on Image Processing*, pages 867–870, Lausanne, September 1996.

[VL97] N. Vasconcelos and A. Lippman. A Bayesian video modeling framework for shot segmentation and content characterization. In *Proceedings of the IEEE workshop on content-based access of image and video libraries*, San Juan, Puerto Rico, June 1997.

[VL99] N. Vasconcelos and A. Lippman. Probabilistic retrieval: new insights and experimental results. In *Workshop on CAIVL, CVPR '99*, Denver, 1999.

[vR79] C.J. van Rijsbergen. *Information retrieval*. Butterworths, London, 2nd edition, 1979. Out of print, available online from http://www.dcs.glasgow.ac.uk/Keith/Preface.html.

[vR86] C.J. van Rijsbergen. A non-classical logic for information retrieval. *The computer journal*, 29(6):481–485, 1986.

[WBKW96] E. Wold, Th. Blum, D. Keisler, and J. Wheaton. Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, 3(3), 1996.

[Wie99] R. Wieringa. A survey of structured and object-oriented software specification methods and techniques. *ACM Computing Surveys*, 30(4):459–527, December 1999.

[Wil93] A.N. Wilschut. *Parallel query execution in a main-memory database system*. PhD thesis, University of Twente, 1993.

[WKSS96] H. Wactlar, T. Kanade, M. Smith, and S. Stevens. Intelligent access to digital video: The Informedia project. *IEEE Computer*, 29(5), May 1996.

[WLM+97] J.K. Wu, C.P. Lam, B.M. Mehtre, Y.J. Gao, and A. Desai Narasimhalu. Content-based retrieval for trademark registration. *Multimedia tools and applications*, 1(3), 1997.

[WNM+95] J.K. Wu, A. Desai Narasimhalu, B.M. Mehtre, C.P. Lam, and Y.J. Gao. CORE: a content-based retrieval engine for multimedia information systems. *Multimedia Systems*, 3:25–41, 1995.

[WS95] M. Wechsler and P. Schäuble. Speech retrieval based on automatic indexing. In *Final Workshop on Multimedia Information Retrieval (MIRO'95)*, October 1995.

[WY95] S.K.M. Wong and Y.Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68, January 1995.

[XC96] J. Xu and W.B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 4–11, Zürich, Switzerland, 1996.

[ZCA97] A. Zhang, B. Cheng, and R. Acharya. A fractal-based clustering approach in large visual database systems. *Multimedia tools and applications*, 1(3):225–244, 1997.

[ZKS93] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1(1):10–28, 1993.

[ZM98] J. Zobel and A. Moffat. Exploring the similarity space. *SIGIR Forum*, 32(1), Spring 1998.

Samenvatting

Een database management systeem (DBMS) is een generiek softwareprodukt dat het mogelijk maakt om gegevensbanken te definiëren en beheren ten behoeve van verschillende toepassingen. Relationele database management systemen worden al jaren met succes toegepast voor het beheer van administratieve data in zakelijke toepassingen.

Dit onderzoek bestudeert het beheer van data in digitale multimediale bibliotheken, en dan met name de implicaties van dit toepassingsgebied op het ontwerp van database management systemen. Het grootste verschil met administratieve toepassingen zit in het verschaffen van toegang tot multimedia gegevens. Omdat de inhoud van administratieve gegevens eenvoudig is te representeren met alfanumerieke waarden, heeft de databasegemeenschap zich geconcentreerd op de rol van de logische structuur van gegevens. Het representeren van de inhoud van multimediale gegevens is helaas niet zo eenvoudig, met als gevolg dat het beheer van die gegevens niet voldoende wordt ondersteund in huidige database management systemen.

Onderzoek naar information retrieval (IR) heeft het mogelijk gemaakt tekstuele documenten terug te zoeken op basis van hun inhoud. Daarnaast heeft onderzoek naar bijvoorbeeld computer visie en beeldanalyse geleid tot zoektechnieken voor verzamelingen van plaatjes en geluiden. Zulke systemen zijn echter vaak losstaande systemen, die slechts enkele zeer specifieke toepassingen kunnen ondersteunen; er is maar weinig consensus over de manier waarop deze technieken in generieke database management systemen geïntegreerd zouden moeten worden. Gangbare oplossingen voegen een beperkt aantal nieuwe functies toe aan de op structuur gerichte vraagtalen, maar deze functies vormen niet meer dan een gemeenschappelijke interface naar nog steeds compleet gescheiden systemen. Dit veroorzaakt twee lastige problemen voor gebruikers van deze systemen: (1) het formuleren van vragen in termen van deze technieken en (2) het samenvoegen van de resultaten op basis van verschillende technieken tot één eindantwoord. Bovendien leidt deze aanpak tot inefficiënties bij het verwerken van vragen die meerdere verschillende abstracties van de inhoud van multimediale gegevens gebruiken.

Net als een gewoon DBMS, is een multimedia DBMS (MM-DBMS) een generiek software systeem dat verschillende toepassingen kan ondersteunen. De geboden functionaliteit richt zich echter specifiek op het domein van digitale bibliotheken. Dit

proefschrift identificeert vier eisen aan zulke systemen: (1) de inhoud van multimediale objecten kan gerepresenteerd worden, (2) het bevragen van de database is een interactief proces, (3) de verwerking van vragen kan gebruik maken van verschillende representaties van de inhoud van objecten, en (4) het formuleren van vragen garandeert de in dit proefschrift geformuleerde notie van content independence. Om aan deze eisen tegemoet te komen, ondersteunen de Mirror architectuur en het Mirror DBMS, een prototype implementatie van deze architectuur, naast het beheer van de logische structuur van gegevens tevens het beheer van abstracties van de inhoud van deze gegevens.

In het ontwerp van het Mirror DBMS vormen het beheer van inhoud en structuur één geheel. Als theoretische basis voor zoeken op inhoud van multimediale gegevens is eerst het 'inference network model' aangepast voor multimedia retrieval. Om het zoeken met dit model te integreren met de verwerking van het zoeken op structuur in database management systemen, wordt in het Mirror DBMS de logische algebra uitgebreid met operatoren voor probabilistisch redeneren. Zo'n volledige integratie opent de weg tot nieuwe technieken voor query optimalisatie, en vereenvoudigt het gebruik van distributie en parallellisme voor de evaluatie van typische IR-vragen.

Andere karakteristieke eigenschappen van multimediale bibliotheken vereisen de ondersteuning van distributie van zowel data als operaties, en de uitbreidbaarheid van het systeem met nieuwe data typen en operaties. Als een oplossing voor deze problemen stelt dit proefschrift voor om het monolithische ontwerp van traditionele database systemen te vervangen door een architectuur op basis van integratie van geavanceerde middleware met database technologie. Dit idee is eveneens uitgewerkt in het onderzoek door middel van de implementatie van een prototype.

De voorgestelde architectuur voor multimedia database management systemen is op drie manieren geëvalueerd, met behulp van de prototype implementatie van het Mirror DBMS. De voordelen van een integrale aanpak zijn geschetst in een serie voorbeeldvragen voor verschillende soorten informatiebehoeften. Het voor dit onderzoek ontwikkelde multimedia retrieval model is getest in enkele kleinschalige experimenten, op een collectie met muziekfragmenten en op een collectie met foto's. Deze experimenten bevestigen het vermoeden dat het redeneren op basis van verschillende abstracties van de inhoud van multimedia gegevens nuttig en mogelijk is. Ten slotte is de snelheid van het systeem getoetst op een grote standaard benchmark voor het zoeken in tekstuele documenten.

Topic Index

Author Index