An OODBMS-IRS Coupling for Structured Documents

Marc Volz, Karl Aberer, and Klemens Böhm
GMD-IPSI, Dolivostraße 15,
64293 Darmstadt, Germany,
{volz, aberer, kboehm}@darmstadt.gmd.de

Abstract

Requirements of modern Hypermedia Document Systems include support for structured documents and full DBMS and IRS functionality.[1] *An objective of DBMSs is to store and facilitate updates of highly structured and typed data. In conventional IRSs, however, documents are perceived as flat. Uncertainty, on the other hand, is a principal notion in the IRS context. In order to combine the advantages of both DBMSs and IRSs we have coupled the IRS INQUERY with the OODBMS VODAK. To model document structure, we have chosen SGML. SGML documents are stored in the OODBMS, and additional full text indexing is done by the IRS. The coupling consists of OODBMS classes encapsulating the IRS functionality. As SGML elements are modeled by database objects in an object oriented framework, queries can be modeled by means of methods—including content based queries.*

1 Introduction

Due to the proliferation of information highways and digital libraries, administering very large hypermedia document bases is becoming more and more important. Existing systems do not yet provide all the functionality needed by the information systems of the future. As an example, consider the MultiMedia Forum (MMF) [Sue+94], an interactive online journal from our institute. There are different ways in which articles of this journal may be accessed: via the table of contents of a particular issue, by means of navigation, or by means of specific database queries. In more detail, queries may refer to document characteristics, e.g., all press releases may be selected. As MMF-documents are SGML documents ([Bry88]) that conform to a proprietary document-type definition, this is feasible. Database functionality is essential, because the document stock may be modified by the editors concurrently with other access operations. Moreover, one would like to be able to formulate one's information needs with a certain degree of vagueness with regard to document content. The content may be either in textual form or of a continuous media type.

In this article, we describe the approach we have taken to build a system incorporating these characteristics. Some of the concepts described here are dealt with in [VAB96] in greater detail.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1]In this article, we use the following abbreviations: DBMS = Database Management System, IRS = Information Retrieval System, OODBMS = Object Oriented Database Management System, SGML = Standard Generalized Markup Language.

1.1 IRS and OODBMS Features

In this section we briefly summarize some of the main IRS and OODBMS features relevant for Hypermedia Document Systems. IRSs manage sets of independent documents called "collections". In most IRSs documents are seen as a list of words. No inner structure (e.g., hierarchies of chapters, sections, and subsections) is supported. Collections are mapped to an index structure (often an inverted list of words), which is usually stored within the file system. Indexing is a complex process, where, e.g., word stemming and thesauri are used. An IRS query usually consists of terms (words) and is against the documents in a collection. During query processing, the IRS computes relevance values for each document. The result is a ranked list of documents that are supposed to be relevant to the query. A central aspect of IRSs is that uncertainty is taken into account in index structures, in queries, and in the matching process during query processing.

OODBMSs can store highly structured data. According to [Atk+89], features of an OODBMS are persistence, concurrency control, recovery, and declarative access (from the DBMS perspective); complex objects, object identity, encapsulation, types and classes (including inheritance), and extensibility (from the OO perspective).

1.2 Analysis of Hypermedia Document System

The following properties should be supported by a hypermedia document management system:

(1) Support for structured documents: The document model should include support of hierarchies and hyperlinks.

(2) Support of full DBMS functionality.

(3) Support of IRS functionality: Regular expression search in text documents is not sufficient.

With regard to (1), document standards such as SGML [Bry88], HyTime, ODA, etc., have been developed. With regard to (2), DBMSs are commercially available. OODBMSs allow management of documents with complex structure [BAN94], such that (1) and (2) are addressed simultaneously.

When considering requirement (3), there is an important difference from the other requirements. Standardization, which is a major issue with the first two cases, is not possible in this particular context. Namely, a formal definition of the semantic interpretation of document content cannot be given. Rather, there is a variety of approaches with regard to automatic interpretation of documents, not just one correct way of extracting information extraction from text. If other media types, such as images, video, or audio, come into the fray, the situation is aggravated. In consequence, when building an integrated system, the architecture must be flexible enough to support eventual replacement of the retrieval component. Combining the three basic requirements leads to a further important property.

(4) Integration of the features mentioned so far. While all of requirements (1), (2), and (3) should be met, this should not give rise to unnatural restrictions on a logical level. Beyond that, efficiency of the implementation should not be traded for full integration on the logical level. The particular semantics of the data model and access operations for more efficient processing must be exploited by the system.

1.3 Handling SGML Documents with OODBMSs

The objective of the HyperStorM ('Hypermedia Document Storage and Modeling') project is to build an object-oriented database application framework for structured document storage [ABH94, BAN94].

Documents' database-internal representation reflects their logical structure. In principle, there is a database object corresponding to each logical document component (e.g., one object corresponding to the title-element, one to the abstract, another one to the introduction, etc.). The database objects corresponding to a document's elements make up a hierarchy. Text or raw data of other media types is contained in the leaf objects. For each element type from the DTD, there is a corresponding so-called element-type class. Its instances are the database objects corresponding to the respective elements. With our database application, there is no restriction to a particular set of document types; rather, documents of arbitrary types can be administered without giving up the element-type information. In [ABH94] it is described how to insert a document-type definition and corresponding documents into the database.

2 Our Coupling Approach

We have applied our coupling approach to the SGML framework described in the previous section. The integration of an IR-component with the OODBMS requires careful design of the interface between the IR component and the OODBMS. On the one hand, it must not restrict the generality of the approach, on the other hand, it must allow for efficient implementation.

Our approach allows the creation of arbitrary IRS collections, e.g., a collection which consists of paragraphs, whereas another collection consists of sections. Object-oriented mechanisms are used to obtain a database object's textual representation and relevance values.

Combined structure- and content-oriented queries are done within the OODBMS query language.

2.1 General Design Decisions

We have chosen to focus on a *loose coupling* between the OODBMS and the IRS for the following reasons: a) it allows for a greater flexibility with regard to the IRS paradigm used; and b) it can be implemented with less effort. The advantages of a tight coupling are a) increased performance, b) higher degree of concurrency, and c) easier handling of updates.

Another important decision has been to let the OODBMS be the control component. The application programmer has access only to the OODBMS. Access to the IRS is indirect via the OODBMS. The advantages are: a) full OODBMS functionality, e.g., with respect to the query language, does not have to be reimplemented; and b) kernel manipulations in both the OODBMS and IRS are avoided.

The overall architecture is shown in Figure 1. The application-, SGML-, and coupling schema are OODBMS schemata. An OODBMS schema consists of class definitions. Information about the SGML schema can be found in [ABH94]. It defines the framework for storing SGML documents. The application schema models the general environment for documents, e.g., document containers and application semantics. A short overview of the coupling schema is given in the next section.

2.2 The Coupling Schema

The coupling schema essentially provides two classes: IRSObject and COLLECTION. Class IRSObject is used as a base class for each application specific class whose instances may be subject to content-oriented queries (on their textual representation). Instances of the database class COLLECTION correspond to one IRS collection. This is similar to [HaW92]. The number of IRS collections in use is arbitrary. Two basic methods of COLLECTION are:

- indexObjects(specQuery: DBQuery) creates an IRS index using the text of the database objects which are specified through database query specQuery.

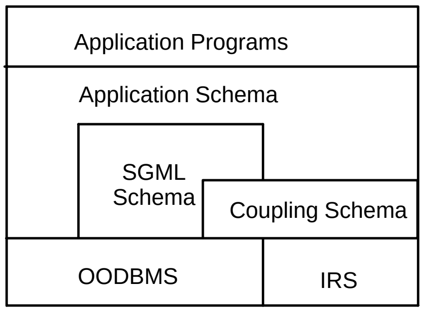

Figure 1: Building Blocks of the Coupled System

- findIRSValue(IRSQuery: STRING, obj: IRSObject) returns the IRS value for the parameter object with regard to the parameter query.

Each document element type is a subclass of the database class IRSObject. Basic methods of IRSObject are:

- getText() returns an object's textual representation. It is used by indexObjects; its implementation is application-specific and must be provided by the application programmer.
- getIRSValue(c: COLLECTION, IRSQuery: STRING) returns the relevance value of the target object for a given IRS context (a document collection) and an IRS query.

2.3 Creation and Usage of IRS collections

As we have described, SGML documents are stored in the OODBMS. On a logical level, each SGML element is represented by one database object. To incorporate IRS functionality, IRS collections have to be created from the OODBMS data. In other words, a mapping from database objects to IRS documents is needed.

One purpose of the mapping is the assignment of relevance values (returned by the IRS as a result of an IRS query) to database objects. An efficient strategy is to assign exactly one IRS document to each database object. Most IRSs, including INQUERY, allow the storage of attributes along with the IRS document. We use that feature to store the object identifier (OID) of a database object directly with the IRS document. Then it is easy to retrieve the OIDs from the IRS documents.

The second purpose of the mapping relates to the creation of the IRS collection. Within an object oriented environment, it is easy to provide each database object with the functionality to create 'its' IRS document. The open question is: Which parts of possibly large documents should become IRS documents? Some answers are (see [VAB96]):

- Each SGML document becomes an IRS document. The disadvantage is that granularity is coarse with potentially big documents. No information can be obtained about the relevance of document elements, e.g., chapters or paragraphs.
- Each document element of a specified element type becomes an IRS document. This approach is used in most coupling approaches, e.g., [CST92, GTZ93].
- Each leaf node of a document becomes an IRS document (finest granularity).

```
<MMFDOC>
        <LOGBOOK> ... </LOGBOOK>
        <DOCTITLE>Telnet</DOCTITLE>
        <ABSTRACT></ABSTRACT>
        <PARA>Telnet is a protocol for ...</PARA>
        <PARA>Telnet enables ...</PARA>
</MMFDOC>
```

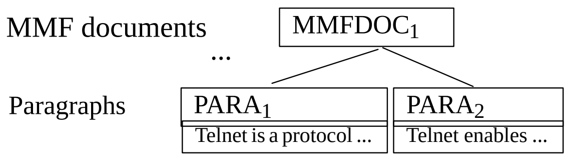

Figure 2: Sample SGML Document and its Internal Representation

- One might want to have IRS documents of approximately the same size [Cal94].
- In some cases, one might want to support the processing of certain query types. For instance, this might be accomplished by choosing a fine-grained granularity for documents or document components written by authors which are referenced frequently.
- An objective may be to keep update mechanisms simple.

If one wants to furnish each document element with IR functionality, problems occur with hierarchically structured text. Consider the fragment in Figure 2 of an MMF document. On the right hand side the logical structure of this fragment is depicted, which essentially is the database-internal representation: If indexing is at the document-level (the complete text of each document becomes an individual IRS document), content-based queries referring to individual paragraphs cannot be evaluated. This can be avoided by additionally inserting the textual representation of each paragraph into the IRS collection. Then, however, the same text is stored at least twice within the IRS.

A general solution for the mapping problem cannot be given. However, the application programmer can be furnished with a flexible mechanism to define the granularity of IRS documents. The mechanism consists of two steps: identifying the database objects (IRSObject instances) which should be represented within an IRS collection, and specifying the textual representation of each selected database object.

Identifying the database objects is done through an arbitrary database query (the *specification query*), which is a parameter of the COLLECTION instance method indexObjects. COLLECTION instances are created by the application. They are used as retrieval context during query processing. It heavily depends on the application semantics which objects are selected to make up a collection - some criteria were given above.

The textual representation of each database object is identified by the getText method of IRSObject, which has to be implemented by the application programmer. getText is invoked for each object identified by the specification query. The results are stored within a file, which is indexed by the IRS, making up the IRS collection. The application programmer is free to decide which text is returned by getText: it could be the complete text stored in properties of the IRSObject instance, parts of that text, manipulated text, or even text which is derived from related objects. To provide more flexibility, the getText method is parameterized, so that the textual representations can depend on a given parameter, and different representations can be defined for different collections.

The specification query and the implementation of the getText methods determine the degree of text redundancy within the IRS collections. If the application programmer decides to create two collections, and the textual representations overlap, there will be redundancy in the IRS index. To avoid redundancy our approach supports the derivation of relevance values from known values, so that a collection at the finest level (normally the leaf objects within an object hierarchy) will be sufficient. This is described in full detail in [VAB96].

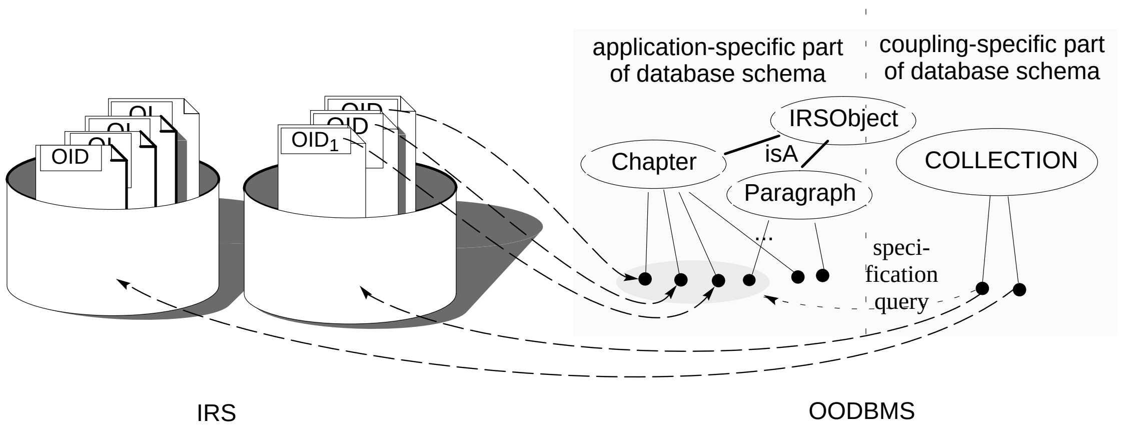
application-specific part of database schema
coupling-specific part of database schema
IRSObject
isA
Chapter
Paragraph
COLLECTION
...
speci-fication query
OID
OID_1
IRS
OODBMS

Figure 3: Modeling/Architecture Juxtaposition

The result of the indexObjects() method call is an IRS collection. With each IRS document the object's OID is stored. On the right hand side of Figure 3 the OODBMS content is shown (shadowed rectangle). It consists of database classes (ellipses), including the coupling classes IRSObject and COLLECTION. Domain specific classes are Chapter and Paragraph, which are derived from IRSObject. Database instances are depicted as bullets. The result of the specification query of one COLLECTION instance is represented by the shadowed area, which includes three Chapter instances and one Paragraph instance. The textual representations of those four objects are stored in the IRS collection on the left hand side of the figure.

2.4 Examples of Mixed Queries

Mixed queries combine structural and content oriented access. The framework is given by the OODBMS query language. VODAK supports OQL (Object Query Language), which is an object oriented extension of SQL. The IRS query part is formulated as arguments of the IRSObject method getIRSValue. The examples are based on MMF documents. collPara is a collection of MMF paragraphs.

"Select all paragraphs and their lengths having an IRS value greater than 0.6 for IRS query 'WWW' ":

```
select p, p -> length() from p in PARA
where p -> getIRSValue (collPara, 'WWW') > 0.6;
```

"Select the title of each MMF document created in 1994 that contains a paragraph element relevant to 'WWW', immediately followed by one relevant to 'NII' ":

```
select d -> getAttributeValue ('TITLE') from d in MMFDOC, p1 in PARA, p2 in PARA
where d -> getAttributeValue ('YEAR') = '1994' and
p1 -> getNext() = p2 and
p1 -> getContaining ('MMFDOC') = d and
p1 -> getIRSValue (collPara, 'WWW') > 0.4 and
p2 -> getIRSValue (collPara, 'NII') > 0.4;
```

3 Application Issues

First we indicate how the coupling approach can be applied to non-textual data. Improved query expressiveness and the application to the WWW are issues we are currently dealing with.

3.1 Applying the Coupling Approach to Non-Textual Media Types

Though we have focused on the problems of structured text documents, our coupling is not limited to those. A practical approach to facilitate information retrieval from images or other multimedia data in documents, for instance, is to have as IRS documents the text fragments that reference the image [CrT91], [DuR93]. The method getText for image objects would return exactly this text.

To give another example, consider an audio segment which can be mapped to text by a speech recognition program. This text can be used for retrieval. The getText method derives this text during the indexing process.

It is even possible to use the same approach for coupling retrieval systems for non-textual data. For example, for image data, an IRS that can build image indexes can be used. The database system passes an image (instead of a text) to the retrieval system, which returns a set of matching images.

3.2 A New Kind of Query Formulation and Evaluation?

The coupled system (ideally) allows the user to retrieve the relevance value of each object of his/her modeling domain (as assumed by the IRS), and to use an arbitrary way of combining relevance values with each other and with structured data. Particularly if hierarchical structures are modeled, this opens opportunities which go beyond those of current IRSs.

Consider a user who is searching for a description of how to use FTP with a WWW browser. With a standard IRS, the user has to submit terms like "WWW" and "FTP" as a query. The result is a list of IRS documents. Matching documents might include 30 page documents which describe network problems (one chapter about WWW, another about FTP) and which do not match the information need of the user. The following query is more likely to select relevant documents; it can be formulated with our system: Return paragraphs relevant to "FTP", which belong to a chapter (or another container object) relevant to "WWW". Although the user can formulate such queries with our coupling approach, a formal model for this kind of query is an open issue. In standard IRSs the combination of terms (e.g., boolean or probabilistic) is possible and is logically evaluated against single documents. In contrast, our approach allows one to combine different query terms with structural features. A query can be given as a pattern for a relevance graph that represents a document structure where the nodes are related with individual IR queries. The query result is not only a single relevance value for each retrieved document (like in IRSs), but a relevance graph. Those IR queries are evaluated against whole document structures or substructures. Similarity matches between query graphs and document graphs is an interesting open issue.

Our future work is to formalize the queries on hierarchically structured documents, and to define or evaluate an appropriate semantics.

3.3 The World Wide Web

Searching for information in the WWW is a difficult task. Numerous search engines try to support it. Most of those search engines are based on simple pattern matching algorithms and do not achieve the functionality of modern IRSs. Moreover, most search engines have a very restricted query interface: complex queries concerning the content and the structure of Web pages cannot be formulated.

WWW server data usually consists of HTML files, stored in the file system. Unfortunately, HTML tags do not have appropriate semantics. In many cases, the tags are used for layout reasons, and do not carry any particular semantics. This severely restricts the benefit of using structural features when searching documents. On the other hand, we see that database solutions become available on the WWW. HTML documents are generated from an internal document representation (e.g., SGML) on-the-fly, providing consistent document structures for large document collections.

Using an expressive query facility might help in the (well-known) case of re-finding information. Many people know *what* HTML document they are searching for, and can often remember some subtle layout details (e.g., "there was a list of at least five bullets, and the second item was bold"). Here the structural search can enhance content-oriented search in a useful manner. That kind of application is independent from the consistency of HTML tags.

In contrast to the well-known centralized search engines (e.g., Lycos, Alta Vista), we plan to design and implement a non-centralized search engine, where each WWW server handles its own index. We see the following benefits from a non-centralized search engine: a) the documents of all participating WWW servers are covered; b) the index is always up to date; and c) distributed query processing is enabled. An example of an established information system using non-centralized indexes is Hyper-G ([AKM94]). Nevertheless, there remain important open issues with non-centralized search engines, e.g., the query must be distributed to several WWW servers. Index buffering might be necessary if queries are against a large number of WWW servers. Those indexes might be installed on Internet hosts. A drawback of index buffering is increasing index update costs.

Important parts of such a non-centralized search engine are realized by our SGML database. A prototype exists, connected to the WWW, including a WWW search interface. We are investigating mechanisms for distributed querying and index replication which might improve performance.

4 Conclusions

In this article we have described our approach to realize a system providing the integrated functionality of an OODBMS and an IRS. In the introduction, we have argued that there should be some flexibility with regard to the retrieval paradigm, and this is best met with a loose coupling. Likewise, instead of the OODBMS VODAK that we have used for the implementation, another OODBMS having the characteristics described in [Atk+89] can be used. We have pointed out the problems that arise when such a coupling is realized. Further, we believe that our approach is rather flexible, for the following reasons: (1) Arbitrary database objects making up an IRS collection can be chosen. (2) The text forming a database object's textual representation can be chosen freely. (3) The way a database object's relevance value is derived from other objects' relevance values can be defined freely.

The following issues remain for future investigation: Relevance feedback has not yet been examined. Uncertainty is not yet adequately considered within the database component of the coupling. Beyond that, we are interested in finding appropriate formulae for calculating an object's relevance value from its subobjects' relevance values. We believe the formula depends on the underlying retrieval paradigm.

Finally, the issue of query optimization in this particular context has to be investigated. As the information-retrieval component is quite fast, in principle it seems worthwhile to generate query-evaluation strategies where information-retrieval subqueries are evaluated first. However, rules for the query optimizer and an appropriate cost model have not yet been specified.

References

[ABH94] K. Aberer, K. Böhm, C. Hüser (1994): "The Prospects of Publishing Using Advanced

Database Concepts", Proceedings of Conference on Electronic Publishing, April 1994, C. Hüser, W. Möhr, and V. Quint, eds., pp. 469-480, John Wiley & Sons, Ltd., 1994.

[AKM94] Keith Andrews, Frank Kappe, Hermann Maurer (1994): "The Hyper-G Network Information System", Information Processing and Management, Special issue: Selected Proceedings of the Workshop on Distributed Multimedia Systems, Graz, Austria, Nov. 1994.

[Atk+89] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, S. Zdonik (1989): "The Object-Oriented Database System Manifesto", Proceedings DOOD, Kyoto, December 1989.

[BAN94] K. Böhm, K. Aberer, E.J. Neuhold (1994): "Administering Structured Documents in Digital Libraries", Advances in Digital Libraries, N.R. Adam, B. Bhargava, and Y. Yesha, eds., Springer, Lecture Notes in Computer Science, 1995.

[Bry88] M. Bryan (1988): "SGML: An Author's Guide to the Standard Generalized Markup Language", Addison-Wesley.

[Cal94] J.P. Callan (1994): "Passage-Level Evidence in Document Retrieval", SIGIR '94.

[CrT91] W.B. Croft, H.R. Turtle (1991): "Retrieval of complex Objects", Proceedings of EDBT 92 (S. 217-229), Berlin, Springer-Verlag.

[CST92] W.B. Croft, L.A. Smith, H.R. Turtle (1992): "A Loosely-Coupled Integration of a Text Retrieval System and an Object-Oriented Database System", SIGIR '92.

[DuR93] M.D. Dunlop, C.J. van Rijsbergen (1993): "Hypermedia and Free Text Retrieval", Information Processing & Management, Vol. 29, No. 3, pp. 287-298, 1993.

[GTZ93] J. Gu, U. Thiel, J. Zhao (1993): "Efficient Retrieval Of Complex Objects: Query Processing In a Hybrid DB and IR System"

[HaW92] David J. Harper, Andrew D.M. Walker: "ECLAIR: an Extensible Class Library for Information Retrieval", The Computer Journal, Vol. 35, No. 3, 1992.

[HeP93] M.A. Hearst, C. Plaunt (1993): "Subtopic Structuring for Full-Length Document Access", SIGIR '93.

[SAB93] G. Salton, J. Allan, C. Buckley (1993): "Approaches to Passage Retrieval in Full Text Information Systems", SIGIR '93.

[Sue+94] K. Süllow et al. (1994): "MultiMedia Forum—An Interactive Online Journal", Proceedings of Conference on Electronic Publishing, pp. 413-422, John Wiley & Sons, Ltd.

[VAB96] Marc Volz, Karl Aberer, Klemens Böhm (1996): "Applying a Flexible OODBMS-IRS-Coupling to Structured Document Handling", ICDE '96.