Published In: 12th International Conference on Data Engineering (ICDE), New Orleans, USA, 1996

Applying a Flexible OODBMS-IRS-Coupling to Structured Document Handling

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

Abstract

In document management systems it is desirable to provide content-based access to documents going beyond regular expression search in addition to access based on structural characteristics or associated attributes. We present a new approach for coupling OODBMSs (Object Oriented Database Management Systems) and IRSs (Information Retrieval Systems) that provides enhanced flexibility and functionality as compared to coupling approaches reported from the literature. Our approach allows to decide freely to which document collections, that are used as retrieval context, document objects belong, which text contents they provide for retrieval and how they derive their associated retrieval values, either directly from the retrieval machine or from the values of related objects. Especially, we show how in this approach different strategies can be applied to hierarchically structured documents, possibly avoiding redundancy and IRS or OODBMS peculiarities. Content-based and structural queries can be freely combined within the OODBMS query language.

1 Introduction

With the advent of information highways and digital libraries the issue of managing and accessing huge hypermedia document bases becomes a core issue. Examples like the MultiMedia Forum (MMF) [Sül+94], an interactive online journal developed at GMD-IPSI, show that existing systems cannot fully support the requirements of such applications. The reader of such a journal may either access a document by means of a particular issue's table of content, by following hypertext links, or by database queries, based upon certain characteristics, e.g., all travel reports. This is feasible because MMF-documents are SGML documents (Standard Generalized Markup Language, [Bry88]) conformant to a proprietary document type definition. Database features are mandatory: for instance, the editorial team may add or modify documents or document components at any time. With this information service, however, it would also be advantageous to allow for formulating information needs with a certain degree of vagueness by accessing the textual and multimedia contents of the documents.

1.1 IRS and OODBMS Features

State-of-the-art IRSs administer sets of independent documents. We assume that an IRS document is a flat text (a list of words). Each document set is called "collection". During the indexing process, the documents within an IRS-collection are transformed to an internal representation (e.g., inverted lists), which are stored in a file system. A number of different approaches have been developed aiming at the extraction of information from documents' content. IRS-queries are given by terms (words) and are against the IRS-documents within an IRS-collection. The result is a set of documents that are likely to cover the information need, often together with an IRS value which indicates the supposed relevance of each IRS document. A central aspect of IRSs is uncertainty of internal document representation, query representation and the matching process during query processing.

In contrast, OODBMS can store highly structured data. Uncertainty is usually not considered. Important features 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) ([Atk+89]).

1.2 Requirements on a Hypermedia Document System

From the analysis of applications the following properties can be derived, that ideally need to be supported by a hypermedia document management system.

(1) Support for structured documents: Hypermedia documents may be structured hierarchically as well as by means of arbitrary hypertext links. The document structures must be freely definable and the underlying system must support access according to these structures.

(2) Support of full DBMS functionality.

(3) Support for content-oriented access: appropriate techniques for accessing the textual and multimedia contents must be supplied. Purely syntactical approaches, like regular expression search in text documents, is not sufficient. Rather, semantic concepts have to be considered.

Existing concepts for each of the three aspects can be characterized as follows. With regard to (1), document standards like SGML [Bry88], HyTime, ODA etc. have been developed. Many tools like editors, document management systems, viewers etc. are based on such standards or document models that are comparable to such standards. With regard to (2), the concepts of DBMSs have been developed for managing structured data in multi-user environments. With advanced DBMS systems like object-oriented ones, it also becomes feasible to successfully manage complex document structures within DBMSs [BAN94], such that the aspects (1) and (2) can be covered simultaneously.

With regard to (3), the situation is somewhat different. Concepts cannot be standardized to the same degree as for the other two cases. The semantic interpretation of document contents cannot be straightforwardly formalized, as it is possible with document or data structures. Thus, many fundamentally different approaches are possible for the interpretation, i.e., information extraction, of textual documents. This situation becomes even more complicate when multimedia contents, like pictures, audio or video, are considered. Therefore, an integrated architecture should allow some flexibility with regard to the retrieval component(s) used.

Combining the three basic requirements leads to a further important property.

(4) Fully integrated usage of the functionalities: This is an aspect that is related to data independency. The availability of the different functionalities must not lead to unnatural restrictions for the user on a logical level. On the other hand, the full integration on the logical level must not sacrifice an efficient implementation, i.e., on a physical level, the system must exploit the particular semantics of the data model and access operations for improved processing.

1.3 Our Approach

The system we have developed takes as starting point an existing application framework for managing SGML resp. HyTime structured documents in an object-oriented DBMS. The main task is the integration of an IR-component with the OODBMS in such a way that it avoids the deficiencies with regard to the integrated functionality of many existing approaches. In particular, this requires a careful design of the interface between the IR component and the OODBMS, which, on the one hand, does not restrict the generality of the approach, but, on the other hand, is efficient. In particular, the following properties are supported by our approach:

- specification of arbitrary (potentially overlapping) document collections that serve as context for content-oriented retrieval queries.
- object-oriented access to text contents and relevance values of objects, i.e., the text of each (document) object can freely be specified and can return its associated relevance value with regard to a query and a document collection.
- provision of a mechanism to combine relevance values that can be used to reduce redundant indexing in collections of hierachically structured documents.

Finally, we deal with the question how such a system can be used to combine structure- and content-oriented retrieval within the OODBMS query language and how such queries can be efficiently processed by the system. The approach can easily be extended to multimedia contents and hypertext document structures.

In Section 2 related work is discussed. Section 3 describes the general architectural framework of our approach. The description of the coupling of the OODBMS with an IR component is contained in Section 4. In Section 5 some indications on the generalization of the approach to general hypermedia documents are provided. Conclusions are in Section 6.

2 Related Work

With regard to related work, a very rough distinction can be made: either an IRS is extended to additionally obtain DBMS functionality or vice versa.

DBMS-Oriented Approaches. With early approaches the relational model has been extended. In [DaD88] the data model ESTRELLA is introduced. It is based on an object-oriented extension of the relational DBMS Oracle.
The "Textual Object Management System" TOMS [DWL92] requires conformance of the full text to a grammar. During query processing, the text is parsed and an index is created. The index can be restricted to specific areas, e.g., to chapters or sections. A similar approach is described in [ACM93].

These approaches have in common that (1) full-text queries are merely regular expressions, (2) results are combined with boolean operators only, uncertainty is not considered, (3) granularity of the textual objects is high, (4) the text itself is not within a database. The DBMS only manages the structure.

IRS-Oriented Approaches. In [HeP93] so called segments are stored redundantly in addition to the documents.

Thus, the hierarchy consists of exactly two levels. [HeP93] and [Cal94] compare three methods of how to split a document into segments: By splitting into equal-length pieces of 30 words, by relying on paragraphs identified by the authors, or by extracting subtopics. Queries are evaluated for the segments, the results are summed up to obtain results for documents as a whole. [SAB93] give up the assumption that complete documents should be retrieved by the IRS. Instead, their system identifies relevant passages of any length and granularity.
[Wil94] has addressed the problem of how to compute IRS values for documents using only knowledge on their parts. SPIDER [Sch93] is introduced as a combined DB/IRS-system, but the database functionality does not yet level up to state-of-the-art database technology. [CrT91] describes the retrieval of complex hierarchical documents with the IRS INQUERY, which is based on Bayesean inference networks. In this work, problems like integration with a complex query language, e.g., integration with structural search or query optimization, have not been considered in detail.

The IRS-oriented approaches have in common that (1) uncertainty and user interaction are part of the framework, (2) texts' structure is not well considered.

Coupling Approaches. COINS [CST92] is the control module for a coupling of INQUERY and the OODBMS IRIS (cf. Figure 1). With that work, it becomes obvious that expressiveness of queries depends on the capacity of the control module. The database schema is a modeling of one particular document type.
With HYDRA [GTZ93], INQUERY and the relational DBMS SYBASE have been coupled. The query is split into an SQL part for structure information and an extended SQL part for full-text information. The first part is passed on to SYBASE, the second part to INQUERY. INQUERY's result, a set of IRS values and IDs, is stored in a temporal SYBASE table, which can be combined with the result of the first part. In [GuN93], SYBASE is replaced by the OODBMS VODAK.
[YaA94] have coupled the OODBMS OpenOODB and TextMachine (TM). TM is a structured text-retrieval system. Documents are only stored in TM, the DBMS just knows the type of a document – and not the structure. Thus, text elements do not have a representation in the database. A basic assumption is that text objects may not be modified. Further, TM does not support uncertainty.
The Extensible Class Library for Information Retrieval (ECLAIR) [HaW92] provides OODBMS classes with retrieval capacities, which are implemented directly without using an existing IRS. Problems occuring with hierachically structured documents are not addressed. Queries are submitted to an OODBMS class IRQuery, which addresses IRS query functionality only.
[SAZ94] introduces problems with managing SGML documents with DBMSs. The authors optimize full text indexing by compression. The objective is to reduce the overhead for multiple indexes on the same data, but different document levels, to about 30%.

Other relevant work has been conducted in the hypertext area. It is not sufficient to see a hypertext node as an independent unit for IR, because the basic assumption that IRS documents are independent is violated. Thus, the link structure must be taken into account [Fuh90, LuZ93].

3 General Architectural Issues of an OODBMS-IRS-Coupling

Loose vs. Tight Coupling

Within a *loose coupling* the cooperating systems can easily be identified. The systems are more independent from each other (they can be used as stand alone systems), there exists a small number of interface routines, and central parts of the data are stored redundantly in both systems. Data has to be interchanged between the systems. On the other hand, with a *tight coupling*, data is shared between the two systems. The individual components cannot be identified at first glance. Often one of the systems is completely integrated within the second system and cannot be used as a stand alone system. This is achieved by implementing and adapting known algorithms within the integrating system in an optimized way.

We have decided to use a loose coupling for the following reasons: Each of the coupled systems can be exchanged more easily, especially if the loose coupling relies as little as possible on peculiarities of the components. Exchangeability enables us to use any kind of retrieval system: e.g. boolean retrieval systems, vector retrieval systems, and systems based on probability. Another advantage of a loose coupling is that it is less costly in development than a tight coupling.

Coordination of OODBMS and IRS

Coordination between the two systems is necessary in order to evaluate mixed queries and to ensure consistency of the data. In this context, we see the following three alternatives:

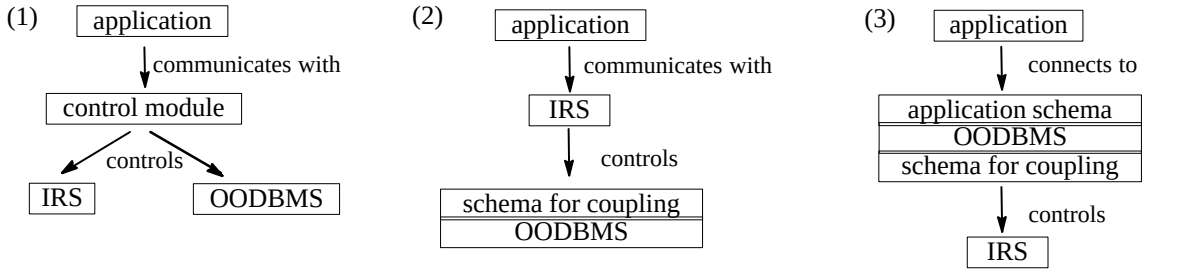


Figure 1: **Different Architectures for a Loose Coupling**

(1) Coordination is accomplished by a third component, which we call *control module* (see Figure 1). In principle, OODBMS and IRS are equivalent. Examples for this architecture are COINS [CST92] and HYDRA [GTZ93]. At first sight, this model seems to be the most flexible one. But we see the following difficulties: (a) A separate component for query processing has to be designed. The expressiveness of the integrated system's query language naturally depends on the capacity of the control module. In case a user interface is developed one has to pay attention that it be not application specific. (b) To facilitate concurrency control for the integrated system, a DBMS-like architecture would have to be developed for the control module.
(2) The application does not communicate directly with the DBMS, but only via the IRS. In this case, we say that *the IRS is the control component*. With this approach there is again the problem that the control component's architecture is not laid out for database functionality. Extending a conventional IRS would require major changes with regard to its architecture.
(3) The DBMS is the control component. The disadvantages of the above approaches can be avoided by using this architecture. The decisive point is that modifying the kernel of any of the existing systems is not necessary. Queries issued by the application, also mixed queries, are expressed in the database query language. Thus, query-processing mechanisms, i.e., analyzing, evaluating and optimizing queries, need not be altered. Formulating complex queries is easy using the database query language. The coupling mechanisms can be provided in a database schema that is, for example, imported into the application schema. Other database features likewise "are for free".

The advantages of the third approach are so enormous that the other alternatives will not be considered any more.

4 Developing a Loosely Coupled System

Based on the basic decision described in the previous section, our approach is now described. As an example for hierarchically structured documents we use SGML documents. An OODBMS schema is used that allows the representation of arbitrary SGML documents. The storage of SGML documents is described in Section 4.1. Additional storage of data within the IRS is controlled by the application.

In Section 4.2 we give a short description of the two OODBMS classes COLLECTION and IRSObject by which the coupling is realized:

4.1 Handling SGML Documents with OODBMSs

At our institute a database application framework to administer structured documents has been realized based on the OODBMS VODAK [ABH94, BAN94]. In the database, documents are fragmented in accordance with their logical structure, i.e., for each element (e.g. section, paragraph, footnote) in a particular SGML document there essentially is a corresponding database object. In other words, each document corresponds to a tree of database objects. Its leaves are the objects that actually contain the raw data, i.e., in most cases, the text. So-called *element-type classes* corresponding to the element-type definitions from the DTDs contain elements of that particular type, i.e., the corresponding database objects. An important feature of our database application is the possibility to manage documents of arbitrary types, i.e., not to be restricted to a rigid set of SGML DTDs. Information on the insertion process of DTDs and SGML documents is in [ABH94].

4.2 Interfaces of Coupling Classes

Instances of database class COLLECTION encapsulate exactly one IRS collection (similar to [HaW92]). The number of IRS collections in use is arbitrary. Each document element is a subclass of database class IRSObject.

Methods of Class COLLECTION

indexObjects (specQuery: DBQuery, textMode: INT, ...): BOOL
indexObjects evaluates the specification query specQuery. The result is a set of IRSObjects. For each of these the method getText (mode: INT): STRING is invoked. The results, in turn, are stored in a file which is indexed by the IRS. To provide different representations of the same IRSObject in different collections, the parameter textMode will be used to distinguish between them.

findIRSValue (IRSQuery: STRING, obj: IRSObject): REAL
The method returns the IRS value for the parameter object. If the object is represented in the IRS collection, the IRS directly calculates the value, otherwise deriveIRSValue (c: COLLECTION, IRSQuery: STRING): REAL is invoked for obj.

getIRSResult (IRSQuery: STRING):
||IRSObject —> REAL||
The IRS query IRSQuery is passed on to the IRS. The result is a dictionary: its keys are the IRSObjects of the text objects, the values the IRS values as computed by the IRS. For both intra- and inter-query optimization, the results of IRS calls are buffered persistently in a dictionary of type ||STRING —> ||IRSObjects —> REAL|| ||. Its keys are IRS queries.

Update methods
One out of three update methods – for insertions, modifications and deletions – has to be invoked whenever a relevant update occurs.

Methods of Class IRSObject

getText (mode: INT): STRING
This method returns an object's textual representation. The argument corresponds to the second parameter of method indexObjects. To allow for different results of getText (mode: INT): STRING for different IRS collections, the method is parameterized.

deriveIRSValue (c: COLLECTION, IRSQuery: STRING, ...): REAL
This method is called whenever an object's IRS value is required, but the object is not represented in the IRS collection. To forward the IRS query to other objects a reference to a COLLECTION instance is needed. The method is invoked automatically if the target object's representation is not in the collection.

getIRSValue(c: COLLECTION, IRSQuery: STRING): REAL
This method returns the IRS value for the parameter query for the target object. It takes two parameters: the document collection that serves as context for the evaluation of the retrieval query expression and the retrieval query expression itself. In essence, it merely consists of an invocation of the method findIRSValue (IRSQuery: STRING, obj: IRSObject): REAL for argument c. getIRSValue is available to ease the formulation of queries (see Section 3.3). From another perspective, with this method each object knows its IRS value, in accordance with the object paradigm.

4.3 Mapping Database Objects to IRS Documents

Generally, an OODBMS stores database objects, whereas most IRSs (e.g., INQUERY [CCH92]) store flat documents. A relationship between the (database) objects and the IRS documents has to be defined for the following two purposes: we have to create IRS documents from the DBMS data, and we have to analyse the result returned from the IRS according to a content based IRS query. To simplify this task, we restrict ourselves to the following relationship: Each IRS document is assigned exactly one object. An object can be assigned to more than one IRS document. The benefit from this restriction is that the mapping of the IRS result to objects is simple and can be implemented efficiently by storing the according object identifier (OID) with each IRS document. This is possible as most IRSs allow to administer some meta data with each IRS document.

The question discussed in the following is how to define the granularity of IRS documents. Some possibilities are:

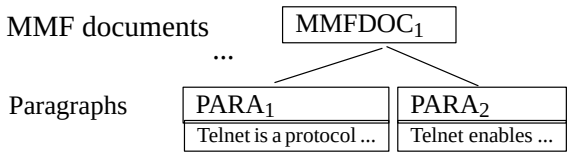
- Each SGML document becomes an IRS document. The disadvantage is that this is a coarse granularity if the documents are big. No information can be obtained about the relevance of the documents elements, e.g., chapters or paragraphs.
- Each document element of a specified element type (the instances of an element class) becomes an IRS document. This approach is used in most known coupling approaches, e.g., [CST92], [GTZ93].
- Each leaf node becomes an IRS document (finest granularity).
- One might want to have IRS documents of approximately the same size [Cal94].
- In some cases, one might wish to both enhance the result quality as well as 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 (see Section 4.6).

One of our objectives is to provide *each* document element with IR functionality. Problems occur with hierarchically structured documents. For illustrative purposes, consider the following fragment of an MMF document:

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

Parts of this fragment are represented within the database as follows:



First, suppose indexing is at the document-level, that is, the complete text of each document becomes a separate IRS document. Thus, content-based queries refering to individual paragraphs cannot be answered. This can be avoided by additionally inserting the textual representation of each paragraph into the IRS collection. Then, however, text is stored redundantly.

4.3.1 Prinicple Solutions for Hierarchical Text

In case both redundancy shall be avoided and explicit querying of small granules shall be facilitated, we see the following alternatives:

(1) create an IRS document for each document element (each DB object), but using a kind of abstract instead of the complete subtext. This abstract can be user-defined (e.g. an introduction of each chapter), or generated automatically (e.g, from the titles of all subobjects),

(2) using compression techniques [SAZ94], or taking advantage of IRS peculiarities (first approach in [CST92]),

(3) inserting IRS documents into IRS collections on the fly before query processing, and deleting them afterwards,

(4) computing IRS values based on known IRS values, e.g., of subobjects ([Wil94] et al).

(1) depends on the peculiarities of the documents (useful abstracts given or not), (2) depends on peculiarities of the IRS, (3) is inefficient due to the fact that inserting and deleting of IRS documents is costly. With ragard to (4), the open issue is how to compute the IRS values of text objects if only components' IRS values are known. An answer must be provided if an object not represented in an IRS collection may be subject to a content-based query. If the paragraphs are represented in the IRS collection, paragraphs' IRS values have to be combined to derive MMF documents' IRS values. We discuss this problem in section 4.5.

4.3.2 Our Solution

At this point it should be clear that the mechanism which defines the granularity of IRS documents has to be very flexible. Within our approach, the granularity is layed down by identifying the IRSObject instances which should be represented as IRS documents through a *"specification query"*, and an IRSObject method which returns the text that is taken for its IRS document. The answer to both questions determines the degree of text redundancy within the IRS collection. Note that with this approach we can realize three of the alternatives presented in 4.3.1, all except (2). Our focus is on (4).

Identifying Text Objects for IRS Handling. The COLLECTION instance is told which objects (IRSObject instances) should be represented in the IRS collection by defining a specification query. The specification query is an OODBMS query expression and thus is powerful enough to specify any reasonable combination of objects. It is the application's task to create a document collection by managing a COLLECTION instance and providing the specification query (see 4.2).

Selecting the objects whose textual representations shall be in the collection essentially depends on the application's semantics, as discussed above in connection with IRS document granularity. In our experiments we have used documents conformant to the MMF DTD.

Which is an Object's Textual Representation? Each IRSObject instance provides the method getText. It is the application programmer's responsibility to implement this method. In this way, arbitrary text fragments can be associated to each database object. The method is invoked by the COLLECTION instance when executing the indexObjects() method to get the object's textual representation for the IRS collection. Within our SGML framework, by inspecting the leaves of the subtree rooted at an element, getText identifies its representation.

The indexObjects() method stores the text of each database object together with the object's OID as IRS documents. Figure 4 shows the resulting relationship of COLLECTION instances and IRS collections as well as the relationship of IRSObject instances and IRS documents (Database classes are represented by ellipses, their instances, depicted as dots, are connected to them with a straight line):

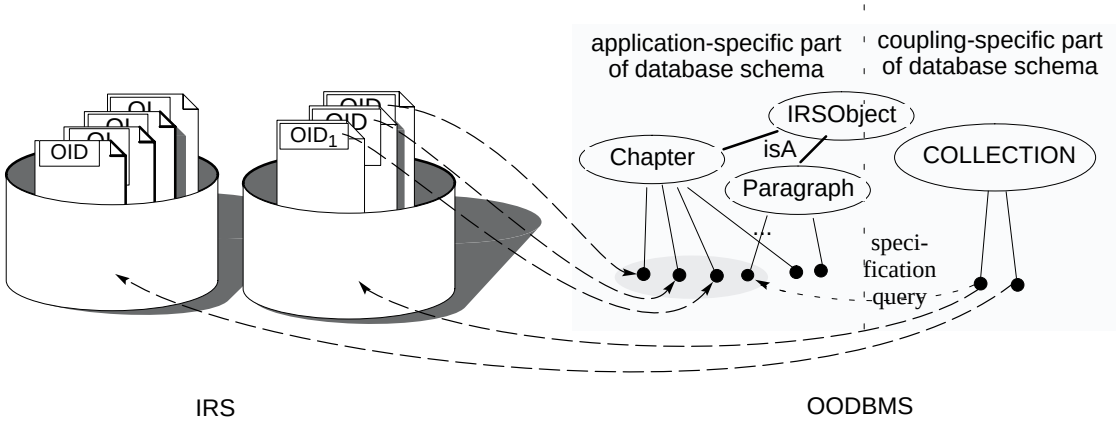


Figure 2: Modeling/Architecture Juxtaposition

4.4 Examples of Mixed Queries

To illustrate the power of our coupling approach, we give examples of *mixed queries*, which consist of both a database and IR part. The main instrument to use content based queries is the IRSObject method getIRSValue (as described in section 4.2). As the query syntax of VODAK is very similar to SQL, we do not describe it in detail. The examples are based on MMF documents. The collection collPara denotes the OID of a paragraph-collection.

"Select all paragraphs and their length having an IRS value greater than 0.6 according to 'WWW'":

```
ACCESS p, p –> length() FROM p IN PARA
WHERE p –> getIRSValue (collPara, 'WWW') > 0.6;
```

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

```
ACCESS      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;
```

4.5 Query Processing

In this section we describe how IRS queries are processed within the database query. The main focus lies on processing content-based queries for database objects which are not represented in the IRS collection.The following flow chart illustrates the processing initiated by calling getIRSValue of an arbitrary IRSObject instance:

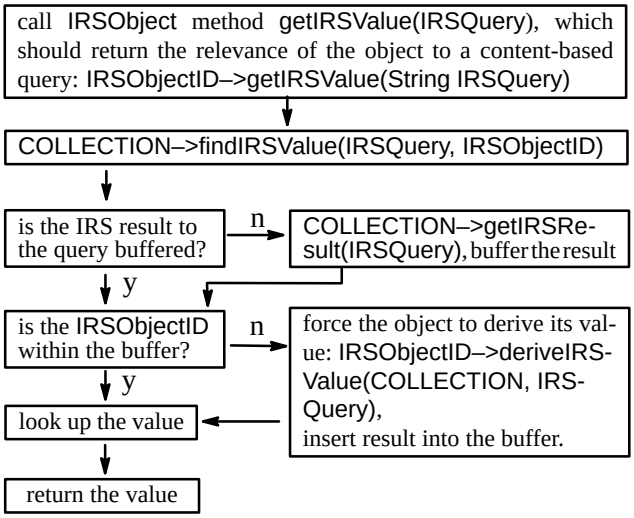


Figure 3: Processing Content Based Queries

The IRSObject method getIRSValue(IRSQuery) determines a COLLECTION instance if not given as an argument, and calls the COLLECTION method findIRSValue(IRSQuery, IRSObjectID). The COLLECTION handles the (external) IRS collection by submitting IRS queries to IRS collections by calling getIRSResult(IRSQuery), which returns for each relevant IRSObject its IRS value. Currently the IRS writes the result to a file which is parsed afterwards to extract the OID-relevance value pairs. This mechanism can be improved by using the API of an IRS. IRS results are buffered to avoid IRS query processing for the same IRS query for different IRSObject instances. If the desired OID is not within the IRS result buffered, the IRSObject instance is

forced to derive its value by calling deriveIRSValue(COLLECTION, IRSQuery), as it will be discussed below.

4.5.1 Choice of Collection Used for IRS Queries

An IRSObject may have different representations in different collections. When content-based queries on such objects are issued it must be decided which COLLECTION to refer to. Therefore, the IRSObject method getIRSValue(IRSQuery) needs to know a COLLECTION instance. This can be achieved (1) by a "hard wired" OID within the method body, (2) as an argument, or (3) as a sophisticated choice of the IRSObject itself.

4.5.2 How to Derive IRS Values?

INQUERY, like most other IRSs, computes IRS values for documents in an IRS collection. A problem occurs if the user wants to evaluate an object whose representation is not part of an IRS collection. In the following we look at the problem for hierarchical documents. Thus, the issue is to provide mechanisms for computing IRS values for objects from its components' values. Some researchers have addressed this problem [Wil94, Cal94, CST92]. Some suggestions are to compute the average or maximum of IRS values of all components [CST92], or to take into consideration the type of the parts, e.g., by weighting the types [Wil94].

With our framework the computation is left open to the application. The application programmer has to decide how derived IRS values should be computed. This is achieved by providing the IRSObjects' method deriveIRSValue. We for our part have run tests with an implementation of deriveIRSValue iterating through the elements components and determining the maximal IRS value.

Consider the MMF documents in Figure 6 together with the relevances for the terms 'WWW' and 'NII':

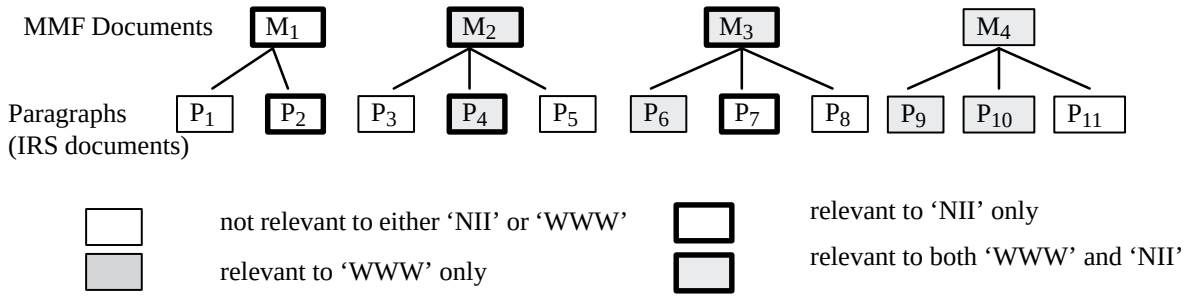


Figure 4: **Structure of Documents, together with Relevance of Elements for Certain Terms**

Suppose that only paragraphs are represented in the collection, that the terms 'WWW' and 'NII' are treated equally by the IRS, and that the paragraphs are of equal length. Consider the query "Select all MMF documents which are relevant to 'WWW' and 'NII'". It cannot be directly answered by the IRS. When redirecting the query to the paragraph IRS documents, the IRS will assign the highest value to P_4, because this is the only IRS document relevant to both terms. To answer the query for MMF documents, an intuitive solution may be to return the MMFDOC objects containing the paragraphs relevant to both 'WWW' and 'NII'. The answer will be document M_2, although M_3 is relevant, too. Hence, good computation schemes combine all components' IRS values, not only highly ranked ones.

Further, with computation schemes such as maximum or average, the query content is not taken into account: MMF documents M_3 and M_4 both contain two 'semi'-relevant paragraphs. Their IRS values, however, should be different, because only M_3 is relevant for both terms. Hence, the information how relevant elements are to the subqueries must be exploited. Hence, first of all, the subqueries need to be identified. Finally, even this information in general is not sufficient to provide a derivation scheme. INQUERY, for example, takes into account the IRS documents' length in order to compute IRS values. Both the component's and the composite's length would be arguments of the derivation scheme. Naturally, there is a variety of other IRS-specific parameters.

With a good derivation scheme, the IRS's effectivity might even be surpassed. Consider the following example: First, assume that IRS documents are to be identified in which a certain term is mentioned at one point. The scheme should be different from the case that documents shall be found in which the term is discussed at great length. Taking such issues into account is future work.

4.5.3 Evaluating Mixed Queries

Consider a query that can be seen as a conjunction of conditions with regard to the structure and to the content The following evaluation alternatives are conceivable:

(1) The query portions are processed independently by the corresponding system, and the results are combined

(e.g., they would be intersected.). This is what we are doing in the sample queries described in section 4.4, which used the IRSObject method getIRSValue. With this approach, restrictions on the search space by the IRS cannot be used by the OODBMS.

(2) The IRS selects all IRS documents fulfilling the conditions on the content. The structure conditions are only verified for the text objects identified in this first step (the opposite approach that the OODBMS restricts the search space for the IRS is not feasible because most IRSs can only search entire collections).
This approach is used in [GTZ93], [HaW92] and others. With our coupling, this approach is also possible. Instead of calling the IRSObject method getIRSValue, the COLLECTION method getIRSResult(IRSQuery) can be used for this purpose.

4.5.4 Optimizing the Evaluation of Mixed Queries

The objective to remain independent of the systems used is difficult to combine with mixed queries' optimization. As the OODBMS is the control component the application communicates with, all queries are expressions in the database query language. On the other hand, IRS-operators can be duplicated as methods of the collection objects. INQUERY's AND-operator, to give an example, corresponds to a method IRSOperatorAND in our implementation. Its parameters are results of IRS queries. Hence, it is possible to calculate conjunction both in the IRS or the OODBMS. Consider the case that the corresponding collection object already knows intermediate results because they have been buffered as the result of previous query evaluations. Then the second alternative is particularly appealing.

Prerequisites for the approach are on the one hand method-based query-optimization features [AbF95], on the other hand a precise knowledge of the IRS-operators' semantics, not only their interface (cf. [YaA94]). For INQUERY, we have knowledge of half a dozen operators' exact semantics. We have implemented them as collection methods to gather experience with optimization issues in this particular setting.

4.6 Propagating Updates

Main features of DBMSs are mechanisms to facilitate updates of the data. With the OODBMS being the control component updates need to be propagated to the IRS. The point of propagation time can freely be chosen within the following bounds: (1) After each database update the corresponding IRS-index structures are updated.(2) After a query is issued the index structures are updated before the query's evaluation.

The first alternative is costly if the number of updates is high as compared to the number of information-need queries. With the second alternative, evaluation of mixed queries is slowed down, because, in general, IRS index structures have to be updated first. From another perspective, the decision when to propagate may either be left to coupling or to the application.

Our realization is as follows: it is the application that invokes the propagation of updates. A good strategy might be to detect low load periods of the system. If, however, an information-need query is issued with update propagation pending, propagation is enforced. Besides that, with some operation sequences, operations cancel out each other's effect. For instance, consider the deletion of a text object that has just been generated. In our implementation, database operations are recorded to avoid unnecessary update propagations, i.e. rebuilding the IRS index structures even though they will not change after all.

5 Applying the Coupling Approach to Non-Textual Media Types

Although we have primarily addressed the problems of hierarchically structured text, our coupling is not limited to this specific field. A practicable approach to facilitate information retrieval from images or other multimedia data in documents, for instance, is having the text fragments as IRS documents that reference the image [CrT91, DuR93]. The method getText for image objects would return exactly this text. To give another example, consider a hypertext-document type containing a binary link type `implies`. The text corresponding to a node shall not only be the physical text of the node. Rather, also the fragments within other nodes' text from which there exists an `implies`-link to that node shall be in the corresponding IRS document. Again, getText would identify this particular text. Moreover, deriveIRSValue can be used to calculate IRS values for hypertext nodes which are not represented in the IRS collection, using the link semantics.

6 Conclusions

This article is a contribution to the ongoing discussion how the integrated functionality of OODBMSs and IRSs can be achieved. An advantage of a loose coupling of existing systems is, among other issues, that there is no confinement to a certain retrieval paradigm. Similarly, VODAK could easily be replaced with another OODBMS meeting the requirements from [Atk+89]. We have described the non-trivial problems occurring with the realization of such a coupling, especially in the field of hierarchically structured documents. We discussed some solutions and showed that our approach can be tailored to most of them. It is possible to realize different solutions with the same framework in parallel and to compare the results which can be achieved. The flexibility of our approach is achieved through (1) precise

specification of DB objects which are represented in IRS collections, (2) the possibility to determine the text used for an objects representation within a collection, (getText method), and (3) the built-in consideration of deriving IRS values for objects from other IRS values of related, e.g. subobjects (deriveIRSValue method).

There are a variety of open issues. Application independent facets are relevance feedback and uncertainty. An unsolved problem is calculating the IRS values for objects using the values for their subobjects. With a good solution to this problem redundant indexing of texts could be avoided. It seems that such an approach depends on the retrieval paradigm the IRS-component is based on (passage retrieval as introduced in [SAB93] seems to be an interesting candidate).

Finally, bringing together the different assumptions ("Open World" vs. "Closed World") is far from trivial. Negation, for example, has a different meaning in both worlds. The semantics of mixed queries including negation remain to be examined, as well as efficient evaluation strategies for such queries.

References

AbF95 K. Aberer, G. Fischer (1995): "Semantic Query Optimization for Methods in Object-Oriented Database Systems", accepted for publication in Data Engineering 1995.

ABH94 K. Aberer, K. Böhm, C. Hüser (1994): "The Prospects of Publishing Using Advanced Database Concepts", in Proceedings of Conference on Electronic Publishing, April 1994, eds., C. Hüser, W. Möhr, and V. Quint, pp. 469-480, John Wiley & Sons, Ltd., 1994.

ACM93 S. Abiteboul, S. Cluet, T. Milo (1993): "Querying and Updating the File", Proceedings of the 19th VLDB Conference, Dublin, Ireland, 1993

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", in: Advances in Digital Libaries, eds., N.R. Adam, B. Bhargava, and Y. Yesha, 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

CCH92 J.P. Callan, B. Croft, S.M. Harding (1992): "The INQUERY Retrieval System", In: Proceddings 3rd Conference on Database and Expert Systems Applications, Sept. 1992

CrT91 W.B. Croft, H.R. Turtle (1991): "Retrieval of complex Objects", In: Proceedings of EBDT 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", 15th Ann Int'l SIGIR'92/Denmark-6/92

DaD88 C. Damier, B. Defude (1988): "The Document Management Component of a Multimedia Data Model", In: Chiaramella, Y. (Hrsg.): 11th International Conference on Research & Development in Information Retrieval, S. 451-464. Presses Universitaires de Grenoble, Grenoble, France

DWL92 S.C. Deerwester, K. Waclena, M. LaMar (1992): "A Textual Object Management System", 15th Ann Int'l SIGIR'92/Denmark-6/92

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

Fuh90 N. Fuhr (1990): "Hypertext und Information Retrieval", working paper, Technical University Darmstadt, Germany

GuN93 J. Gu, E.J. Neuhold (1993): "A Data Model for Multi Media Information Retrieval"

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", ACM-SIGIR'93-6/93/Pittsburgh, PA, USA

ISO86 "Information Processing – Text and Office Systems – Standardized Generalized Markup Language (SGML)", ISO 8879-1986 (E), *International Organization for Standardization*, 1986

ISO92 International Organization for Standardization (1992): "Information Technology – Hypermedia/Time-based Structuring Language (HyTime)", ISO/IEC 10744, 1992 (E)

KAN93 W. Klas, K. Aberer, E.J. Neuhold (1993): "Object-Oriented Modeling for Hypermedia Systems Using the VODAK Modeling Language (VML)", in Advances in Object-Oriented Database Management Systems, NATO ASI Series, Springer Verlag Berlin, eds. A. Dogac et al., August 1994.

Kla+94 W. Klas et al., "VML – The VODAK Model Language Version 4.0", Technical Report, GMD-IPSI, July 1994

LuZ93 D. Lucarella, A. Zanzi (1993): "Information Retrieval from Hypertext: An Approach Using Plausible Inference", Information Processing & Management Vol. 29, No. 3, pp. 299-312, 1993

SAB93 G. Salton, J. Allan, C. Buckley (1993): "Approaches to Passage Retrieval in Full Text Information Systems", ACM-SIGIR'93-6/93/Pittsburgh, PA, USA

SAZ94 Ron Sacks-Davis, Timothy Arnold-Moore, Justin Zobel: "Database-Systems for Structured Documents", First International Conference on the Application of Database Technologies & their Integration, Nara, Japan, October 1994

Sch93 P. Schäuble (1993): "SPIDER: A Multiuser Information Retrieval System for Semistructured and Dynamic Data.", ACM-SIGIR'93-6/93/Pittsburgh, PA, USA

Sül+94 K. Süllow and others (1994): "MultiMedia Forum – an Interactive Online Journal, in Proceedings of Conference on Electronic Publishing", pp. 413-422, John Wiley & Sons, Ltd.

Wil94 R. Wilkinson (1994): "Effective Retrieval of Structured Documents", SIGIR'94

YaA94 T.W. Yan, J. Annevelink (1994): "Integrating a Structured-Text Retrieval System with an Object-Oriented Database System", Proceedings of the 20th VLDB Conference, Santiago, Chile, 1994