Modelling Information Retrieval in an Object-Oriented Database Environment

Michael J. Nelson
Graduate School of Library and Information Science
Elborn College
University of Western Ontario
London, Ontario N6G 1H1
Email: mnelson@julian.uwo.ca

Abstract

Bibliographic or full text information retrieval (IR) applications have been implemented either as special purpose programs or as systems based on database technology. The relational model has been used with some success but most work has used separate IR specific models. The object-oriented data model offers many advantages by incorporating documents as complex objects. Basic concepts such as objects, class hierarchy, inheritance, aggregation, collections and methods are applied to the IR situation. Various approaches to query processing are introduced. To illustrate these concepts a partial implementation in object-Pascal is presented. Problems of standardization and formalization are discussed.

1 Introduction

Most bibliographic or full text information retrieval (IR) applications are implemented as special purpose programs or systems based on inverted file technology. Although there has been much modelling work done in databases, the application of this theoretical data modelling to IR has been scattered. The main reason for this is that the traditional database modelling techniques do not apply easily to IR. In particular most popular database software has been based on the relational model which does not work well for IR applications as explained below. So in most cases IR has been considered an application specific software.

The object-oriented model offers some advantages over the traditional hierarchial, network and relational models. The object-oriented database model was developed primarily for applications with complex objects such as software engineering, computer assisted drafting (CAD) and discrete event simulation. In each of these applications the main focus is on objects which must be manipulated. In software engineering it is programs, functions and procedures; in CAD it is drawings; and in discrete event simulation it is objects in the real world such as cars and traffic lights. The object-oriented approach offers many advantages by providing a more direct and natural description of real-world applications. In IR our focus is on documents, their descriptions and contents which may include programs, sound, pictures (i.e. they may be multi-media). These documents are the central objects of interest in bibliographic and full text information retrieval and qualify as complex objects which can be modelled in an object-oriented environment. As commercial object oriented database management systems become available they can be applied to storage and retrieval problems. Currently though, there is a wide variety of features found in object oriented database management systems so this paper will concentrate on the basic properties agreed upon by most authors and show how these can be applied to a model of bibliographic and full text systems.

2 Previous models used for IR.

2.1 Relational Databases.

Some authors have used the relational database model as a basis for information retrieval. Crawford (1981) showed how bibliographic data could be stored in normalized relations. One of the problems with normalization is that all the data from one bibliographic record is split into many relations. Even the first normal form means that each author is a separate entry in an author table so that assembling a full record is awkward and time consuming. This also makes formulating queries in a query language such as SQL unusually difficult. The only advantage this brings is easier enforcement of data integrity (for example an authority file) and a some flexibility for ad-hoc queries and data manipulation. Another problem is that relational database systems don't provide text and string searching capabilities. Later on Crawford and Yeung (1988) also showed how the traditional Boolean, vector space, probabilistic and fuzzy set models of IR could be implemented using a relational database.

An alternative approach is to extend the relational model with various indexing and text handling functions. This was the approach taken by Schek

(1984) who used a relational database to store strings and had another layer of software to interpret these strings and process queries. Lynch and Stonebraker (1988) also extended the relational model but with an abstract data type which is an encapsulated data structure that is accompanied by a set of user defined functions which manipulate the data type. The user defined functions cover such operations as keyword indexing and searching rules. In object-oriented databases this encapsulation is a basic property and not an extension. Lynch and Stonebraker also added indexing schemes which would allow the specification of inverted indexes.

Macleod (1991) gives a thorough comparison of the relational model to inverted file retrieval systems which he calls 'text models'. He gives a very good description of the characteristics of documents and document collections which cause difficulties in traditional database management systems. To summarize: "A basic problem with the relational model is that it is a record based model and a relatively complex object such as a document does not map naturally into such a model." [Macleod (1991), p.164].

2.2 Other models in IR.

There are various other models for IR not based on DBMS models for example, Boolean, vector space, probabilistic and fuzzy set model. These models concentrate on the indexing methods, query formulation and query processing thus assuming the data structure, storage and indexes is an implementation problem.

The advantage of the object-oriented model is that it encompasses all the data structures, data storage, indexing and querying aspects into a more integrated whole.

Tague et.al. (1991) have proposed an IR model based on a description of the structure of documents defined by a grammar. The access to the documents is defined in terms of a hypergraph model. This is combined into a complete formal model for IR which is more conceptual than most models of IR. Many aspects of this model are also reflected in the object-oriented model.

3 Object-oriented Concepts.

Object-oriented data base concepts and features have come from two main sources: the semantic data modelling approach (Peckham and Maryanski (1988), Tague (1984)) which tried to develop data models with more knowledge about the real world into the data model and object-oriented programming which is concerned with organizing the data and procedures involved in programming for a variety of applications.

3.1 Objects

Before we can develop a full model we need to define some basic terminology used in object oriented applications. This discussion generally follows Bhalla (1991).

An object represents a thing or concept from an application environment. In the IR application it could be a book, article, picture or bibliographic record. It could also be parts which would be assembled into more complex objects. So pictures, authors, tables of contents, etc. could all be defined as objects.

An abstract data type describes the attributes and operations that are used to access or process the object. The combining of all the data and functions into one abstract data type is called encapsulation. This provides data independence by binding the attributes and actions performed on the data together no matter when the objects are accessed. These operations are called methods. In fact the data encapsulated within an object should only be accessed by the methods, so even the display of a simple data element on the screen should have its own method. The attributes may be primitive data types such as integers, more complex data types or even objects themselves.

3.2 Class hierarchy.

A class is the set of objects of one type. Objects that belong to the class are called instances of the class. The documents in a library form a class of objects whereas the book by C.J. Date called 'Introduction to database systems, 5th ed.' is an instance of that class. Classes are organized into a hierarchy called the class hierarchy. A new class, called a sub-class can be created from an existing class by adding additional properties (attributes or operations). For example a monograph type could have a subclass of monograph-in-series which has the additional attribute 'series title' and 'series editors'. A 'monograph-in-series' "is-a" monograph, so these hierarchies are sometimes call "is-a" hierarchies. The objects in the sub-class inherit all the properties of their superclass. So if a document has an attribute called 'title' then all the descendants will also have the attribute 'title'. This means the object-oriented model has all the properties of the hierarchial database model, but of course it has much more. An example of one possible document hierarchy is shown in figure 1 using a graphical notation based on the entity-relationship model (McFadden, 1991). An arrow goes from the sub-class to the superclass.

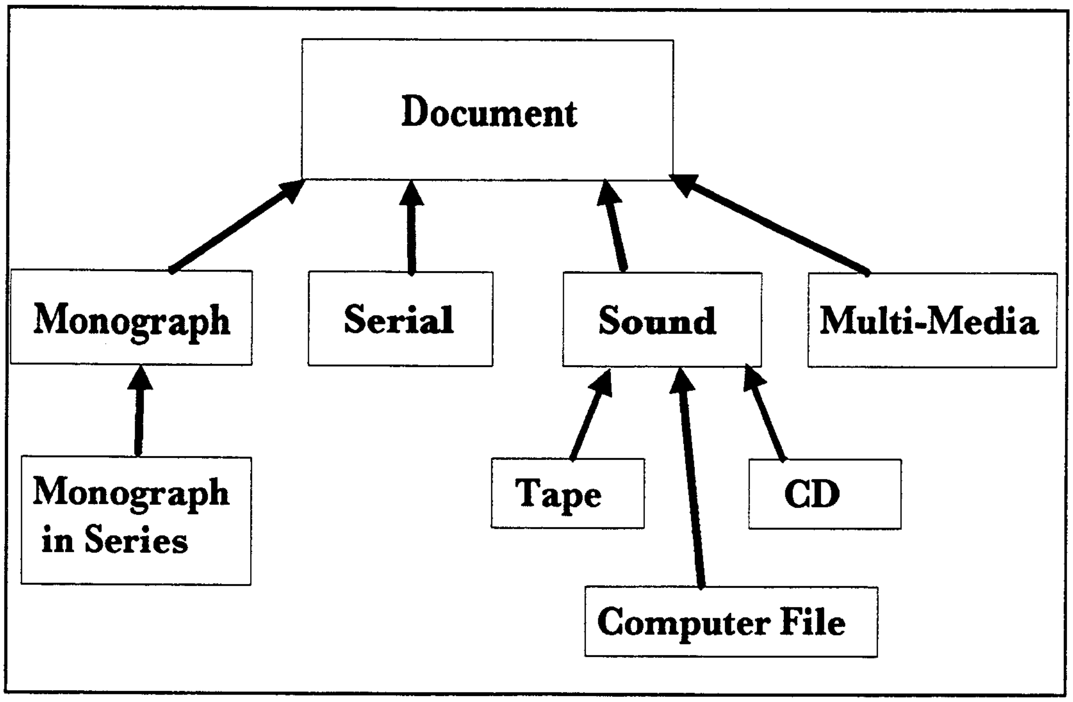

Figure 1

3.3 Aggregation

Another important concept is aggregation where component objects or attributes are put together to form an object.. The components may be attributes such as title or call number but can also be other objects such as pictures. This a powerful way of building complex document objects which then may include many different types of objects such as text, sound and pictures. In fact even an author may be an object. This gives us a "part-of" hierarchy since the components are part of the main object. In many models of full-text this is the essential relationship which is modelled. If 'text' is defined as an object then a paragraph is a "part-of" the 'text'.

3.4 Sets and collections.

In many objects there is also a need for the concept of 'a set of objects' which is called a collection. For example in defining a 'text' object several sentences together form a paragraph, several paragraphs form a section, etc. In a bibliographic description a monograph may have a as its authors a collection of names where a 'name' is also an object. The advantage of a collection is that the number of elements is not fixed so that a document may have any number of authors. An example of aggregation and collections is shown in figure 2, which is based on the book model of Woelk, Kim and Luther (1986). So 'cover' and 'table of contents' are al "part-of" 'book' and 'contents' is an collection of chapters and a chapter is itself a collection of sections. Of course each of these are objects in there own right as are the other components in the book such as table of contents which much each be defined as an object.

Book Object
- Cover
- Table of contents
- Preface
- Contents
  - chapters (Collection of objects)
    - sections (Collection of objects)
      - paragraphs (Collection of objects)
- References
- Index

Figure 2

A better way to approach this model might be to define a general 'text' object which could be used in any type of document. This 'text' type would encapsulate all the data and operations needed to manipulate full text for retrieval purposes and could be based on a general model of text such as that by Desai et.al. (1986).

3.5 Methods and messages.

So far the examples have concentrated on the data. What about the methods? The methods are determined by the functions that are need for information retrieval. A simple example is that we need a method to display each object on the screen and each object can have its own version of the display method which only has to be defined once. The main functions though are indexing and query processing. The indexing rules can be incorporated as methods in the objects, for example if we have a 'name' object there would be an 'index' method (a procedure) which defines how names are entered into the index. Then whenever an object needs indexing a <u>message</u> is sent and the method is automatically executed. Thus the indexing method is only defined once for the general object 'name' and all instances of 'name' can then use that method. Note that a method can be inherited by a subclass or overridden by a new method for that sub-class. This offers flexibility when implementing a database by offering built in indexing rules which can be overridden by the implementor.

3.6 Queries.

Query processing is one area where object-oriented database management systems (OODBMS) are particulary diverse. Some OODBMS use an extended SQL which is based on the relational model. Others have a procedural language for querying objects. If the indexing methods mentioned previously create the indexes as objects then another approach is to write methods to search the indexes. The indexes must then keep track of the object identities and class types so that queries can restrict searches to certain classes such as 'monographs in series'. This approach puts the responsibility for searching on the designer and implementor of the particular database but on the other hand gives them more control over the behaviour of the system. This also points out one of the main advantages of the object-oriented approach which is extensibility. Methods can be extended or overridden very easily with a customized version.

The most general proposals for query processing allow for the manipulation of sets and lists which is very useful for IR. For example the proposed Object Database Standard (Atwood et.al., 1994) allows for four types of collections: sets, bags, lists and arrays.

Another approach is to use an IR front end with an object database used for the storage of documents (e.g. Croft, Smith & Turtle, 1992).

4. Implementation

One of the difficulties in data modelling is separating the conceptual model of the data from the actual implementation. It is very difficult to model data completely independently from the eventual implementation on a computer.

These general models could be implemented in many (but not all) of the OODBMS currently on the market. An alternative is to use an object-oriented programming language such as Smalltalk, C++ or Object Pascal to implement the data model. The disadvantage of using the programming languages is that the standard database operations are not included. Parsaye et.al.(1989, p.145) believe the minimal database capabilities needed are:

"1. A high-level query language with query optimization capabilities in the underlying system.

2. Support of persistence and atomic transactions: concurrency control and recovery.

3. Support of complex object storage, indexes and access methods for fast and efficient retrieval."

The example data model above was implemented in Borland Pascal 7.0 in Windows using the Object Windows Library. This offers some extra features including a method of storing objects in "streams" which are stored in disk files. Thus many objects of several different types can stored in a file and accessed randomly. These storage methods are for a general object called "TObject" which is then the ancestor of all our objects. The abstract "TObject" also contains all the necessary collection objects and methods. An example of some of the main definitions in Borland Pascal for the document hierarchy from figure 1 is shown in figure 3.

```
document = object(Tobject)
   document_id: Pchar;
   constructor init;
   destructor done; virtual;
   procedure display; virtual;
   end;

monograph = object(document)
   title: pChar;
   authors: name_collection;
   .
   .
   .
   constructor init;
   destructor done; virtual;
   procedure display; virtual;
   end;

monograph_series = object(monograph)
   series_title: pChar;
   series_editor: name_collection;
   constructor init;
   destructor done; virtual;
   procedure display; virtual;
   end;
```

Figure 3.

5. Some problems.

One of the main problems when working with object-oriented databases is the problem of standardization. Different commercial products and experimental systems all have difference definitions, capabilities and features. There is a proposed standard which has recently been published (Atwood et.al. (1993)) but it is too soon to see how widely it will be adopted or even if it gains any wide agreement amongst researchers and practitioners.

Another weakness mentioned by Kim (1991) is that the object-oriented databases are not based on a formal language or mathematical model like the relational model. As Kim explains there has been some work to extend the relational model by certain object-oriented constructs and some other research trying to base the object-oriented formalism on first-order logic.

6. Future work.

This model is currently a skeleton of data descriptions which is intended to be the underlying engine for a general retrieval system. This model needs a more complete and detailed analysis with some attempts for completeness. Unfortunately, completeness is the responsibility of the designer and implementor in the object-oriented approach and is more difficult to check than in some other approaches. On the implementation side the code must be filled in and integrated with other work on the user interface and hypertext extensions. Another possibility is to evaluate the feasibility of implementing the model into a commercial object-oriented database system.

On a more theoretical front, a careful look at the grammar based model of Tague (1991) is needed to see if it is possible to integrate the two models or at least incorporate all the necessary functions and descriptions into this model. This should be combined with a more concerted effort to separate the modelling from the implementation in order to develop a conceptual object-oriented model.

BIBLIOGRAPHY

Atwood, T., Duhl, J., Ferran, G., Loomis, M., & Wade, D. (1994). *The object database standard: ODMG-93* (R. G. G. Cattell, Ed.). Menlo Park, CA: Morgan Kaufman.

Bhalla, N. (1991). Object-oriented data models: a perspective and comparative review. *Journal of Information Science*, *17*(3), 145-60.

Blair, D. C. (1988). An extended relational document retrieval model. *Information Processing and Management*, *24*, 364-371.

Christodoulakis, S., Theodoridou, M., Ho, F., Papa, M., & Pathria, A. (1986). Multimedia document presentation, information extraction, and document formation in MINOS: A model and a system. *ACM Transactions on Office Information Systems*, *4*(4), 345-383.

Crawford, R. G. (1981, January). The Relational Model in Information Retrieval. *Journal of the American Society for Information Science*, *31*(1), 51-64.

Crawford, R. G., & Yeung, M. K. (1988). Specifying information retrieval in a relational system. *Canadian Journal of Information Science*, *13*(1/2), 1-16.

Croft, W. B., Smith, L. A., & Turtle, H. R. (1992). A loosely-coupled integration of a text retrieval system and an object-oriented database system. In N. Belkin, P. Ingwersen, & A. M. Pejtersen (Eds.), *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retreival* (pp. 223-232). New York: ACM.

Date, C. J. (1990). *An introduction to database systems* (5 th ed.). Reading, Mass.: Addison-Wesley.

Desai, B. C., Goyal, P., & Sadri, F. (1986). A data model for use with formatted and textual data. *Journal of the American Society for Information Science, 37*(3), 158-165.

Kim, W. (July/August 1991). Object-oriented database systems: strengths and weaknesses. *Journal of Object Oriented Programming, 4*(4), 21-29.

Kitchel, S. W. (1991). Object-oriented databases for libraries and other complex systems. In C. H. Davis (Ed.), *Database management: how much power is enough?* (pp. 15-29). Champaign, Ill.: University of Ill. at Urbana-Champaign. Graduate School of Lib. & Information Science 1991.

Lynch, C. A., & Stonebraker, M. (1988). Extended user-defined indexing with application to textual databases. In F. Bancilhon, & D. J. DeWitt (Eds.), *Proceedings of the Fourteenth International Conference on Very Large Data Bases, Los Angeles, USA, Aug.29-Sept.1, 1988* (pp. 306-317).

Macleod, I. A. (1990). Storage and retrieval of structured documents. *Information Processing and Management, 26*(2), 197-208.

Macleod, I. A. (1991). Text retrieval and the relational model. *Journal of the American Society for Information Science, 42*(3), 155-165.

McFadden, F. R. (1991, September). Conceptual design of object-oriented databases. *Journal of Object Oriented Programming, 4*(5), 29-33.

Peckham, J., & Maryanski, F. (1988, September). Semantic data models. *Computing Surveys, 20*(3), 153-189.

Schek, H. (1984). Nested transactions in a combined IRS-DBMS architecture. In C. J. van Rijsbergen (Ed.), *Research and Development in Information Retrieval, Proceedings of the Third Joint BCS and ACM Symposium, July 2-6, 1984.* (pp. 55-70). Cambridge, UK: Cambridge University.

Tague, J. (1984). A semantic model and schema notation for bibliographic retrieval systems. In C. J. van Rijsbergen (Ed.), *Research and Development in Information Retrieval, Proceedings of the Third Joint BCS and ACM Symposium, July 2-6, 1984.* (pp. 71-93). Cambridge, UK: Cambridge University.

Tague, J. (1986). The semantic model and information retrieval. In *Proceedings, Conference on the Empirical Foundations of Information and Software Science, Atlanta, 1986* (pp. 287-306).

Tague, J., Salminen, A., & McClellan, C. (1991). Complete formal model for information retrieval systems. In A. Bookstein, Y. Chiaramella, G. Salton, & V. V. Raghavan (Eds.), *SIGIR '91, Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval* (pp. 14-20). New York: ACM.

Woelk, D., Kim, W., & Luther, W. (1986). An object-oriented approach to multimedia databases. In C. Zaniolo (Ed.), *Proceedings of SIGMOD '86, International Conference on Management of Data, Washington, D.C., May 28-30, 1986* (pp. 311-325). New York: ACM.