Documents as Arrays

Ian A. Macleod,
Dept. of Computing
and Information Science,
Queen's University,
Kingston,
Ontario.

ABSTRACT

There has been a significant amount of interest recently in the integration of document retrieval with data base management. Most of this work has centred on the relational model. However there are significant defects with the relational approach. In particular, normalisation causes problems. Documents typically contain repeating fields such as author names, keywords, citations and so on. Normalisation causes what was once a logical unit of information to be dispersed across a number of relations. This greatly complicates the retrieval process. Other problems with the relational model include a lack of hierarchy and the inability to retrieve objects of different types "simultaneously".

In this paper we outline the array model. Here objects are represented as array members rather than set members. Further array members may themselves be arrays. This type of information structure seems to more accurately correspond to the structure of documents. We look at the implications of such a model with respect to the query language and we also look at implementation considerations.

Représentation matricielle des documents dans les bases de données relationnelles.

On remarque beaucoup d'intérêt, depuis quelque temps, pour l'intégration du repérage des documents avec la gestion des bases de données. La plupart des études effectuées sur le sujet ont utilisé le modèle relationnel. Il y a toutefois plusieurs défauts importants à l'approche relationnelle, notamment des problèmes de normalisation. De façon générale, les documents contiennent des champs répétitifs, tels les noms d'auteurs, les descripteurs, les références bibliographiques, etc. Le processus de normalisation fait que ces unités logiques d'information sont dispersées dans un certain nombre de relations différentes, ce qui complique considérablement le processus de repérage. Le manque de hiérarchie et la difficulté de repérer simultanément des objets de nature différente sont d'autres problèmes liés au modèle relationnel.

Le présent exposé introduit le modèle matriciel dans lequel les objets sont représentés comme des éléments d'une matrice plutôt que des éléments d'un ensemble. Il est même possible que certains éléments d'une matrice soient eux-mêmes des matrices. Cette façon de structurer l'information semble mieux correspondre à la structure même des documents. L'auteur analyse enfin les diverses implications du modèle sur le langage d'interrogation et aborde quelques considérations pratiques de fonctionnement.

INTRODUCTION

By and large, information retrieval systems are primitive examples of information systems. They have predetermined search strategies, often can only handle one data base at any one time and are generally inflexible. With the evergrowing availability of machine readable bibliographic data and the increasing accessibility of inexpensive hardware, there is a corresponding growth in interest in flexible document retrieval systems.

Recent work in this area has suggested the adaption of existing data base models to document retrieval. Most of these ideas have focussed upon the relational model, [1,2,3,4]. Of the three "traditional" data base models, this has seemed the most attractive mainly because of the completeness of the associated high level query languages such as SQL, [5], as well as a certain naturalness with the underlying tabular data structures. This naturalness derives from the observation that many of the structures in information retrieval map into tables.

At first sight the relational model seems well suited to this type of application since a lot of the underlying information structures in a document retrieval system are tabular - things like dictionaries, stop word lists, indexes, thesauri and so on, and when we view the logical data organisation as a set of tables, then the relational concepts are ideal. However, when we try to extend the application of the model to more structured data, a number of the deficiencies of the relational model became more apparent.

THREE APPROACHES TO RETRIEVAL

In this section, we illustrate the relational approach together with that taken by a more conventional retrieval system, STAIRS, [6], and look at some sample dialogues. We will compare these approaches with our proposed new model, the array model.

STAIRS is a widely available commercial product distributed by IBM. It is an example of a "conventional" retrieval system produced to handle bibliographic data. It is also the system on top of which BRS, [7], a MEDLINE derivative, has been developed. STAIRS permits a document to be made up from up to eight formatted (fixed length), fields and one variable length text field. The ability also exists to partition the text field into named paragraphs. Two retrieval commands are provided. The first, SELECT mode, performs retrieval based on the content of the formatted fields. The second type of retrieval, SEARCH mode, is based on the words contained in the unformatted text field. The results of both retrieval commands can be combined using the usual set operations.

In SELECT mode, we can search for a particular value of a formatted field or a range search can be carried out. In SEARCH mode, searching is based on the presence of a particular word. Further, words can be required to be in a particular paragraph; a boolean combination may be requested; two words may be required to be adjacent or present in the same sentence or paragraph. Various weighting algorithms can be applied to the results of SEARCH mode operations if something other than strictly boolean retrieval is desired.

For example, suppose we had a document consisting of a title, a set of one or more authors, an abstract a set of keywords and a year of publication. The conventional approach here would be to have a single formatted field called "YEAR" and to split the text field into three "paragraphs", (the term is slightly misleading), called, perhaps, TITLE, AUTHORS, KEYS and ABSTRACT.

In STAIRS, this information would roughly be stored as shown in Figure I. The nested box denotes a repeating field.

```
DICTIONARY

+----+-------+
|word|pointer|
+----+-------+
       |
       |
       |     INVERTED FILE
+---------+----------+-----------------------+
|doc_count|word_count|+-----+----+----+------+|
|         |          ||docno|para|sent|wordno||
|         |          |+--|--+----+----+------+|
+---------+----------+---|--------------------+
                         |
     --------------------
    |                       DOCUMENT
+-----------+----------------+          +----+
|doc_pointer|formatted_fields|          |text|
+-----------+----------------+          +----+
    |                                      |
     --------------------------------------
```

FIGURE I - Approximate STAIRS File Organisation

In the relational model, this information would most appropriately be spread over three relations. These might be

DOCUMENT (D#, Title, Abstract, Year);
AUTHORS (Author, D#)
KEYS (Key, D#)

This organisation is illustrated in Figure II.

```
DOCUMENT
+--+-----+--------+----+
|D#|Title|Abstract|Year|
+--+-----+--------+----+
|                      |
|   DOCUMENT Tuples    |
|                      |
+----------------------+


AUTHORS                KEYS
+------+--+            +---+--+
|Author|D#|            |Key|D#|
+------+--+            +---+--+
| AUTHORS |            | KEYS |
| Tuples  |            |Tuples|
+---------+            +------+
```

FIGURE II - Corresponding Relational Organisation

The approach presented here is to suggest a new model, the array model. This is a somewhat radical step but, as will be noted, the model is based upon a generalisation of the relational model, rather than being totally new. Array theory has grown out of attempts to generalise APL, particularly to extend APL to permit non-rectangular arrays. These are arrays where elements of the same array may have different dimensions and types. Trenchard More at IBM has been one of the leading proponents of array theory, [8,9]. The main interest from our point of view is that the basic information structure seems to be a more reasonable model of reality than do the sets of the relational model.

In the array model, the example document we had earlier can be represented as a single array as shown in Figure III.

```
+-----+---------+--------+-----+----+
|Title|+-------+|Abstract|+---+|Year|
|     ||Authors||        ||Keys||    |
|     |+-------+|        |+----+|    |
+-----+---------+--------+-----+----+
```

FIGURE III - Corresponding Array Organisation

Each array element consists of an array of five items - a title, an array of authors, an abstract, an array of keywords and a year. Because non-atomic attributes are allowed there are no normalisation problems. The result of retrieving something from this array is itself an array. The array model is really a generalisation of the relational model. Instead of retrieving tables from tables we retrieve trees from trees.

At this level significant differences between the three approaches are already noticeable. In STAIRS, formatted (non-text) fields are treated quite differently from text fields. Also there is a limit, though not an unduly restrictive one, on the number of formatted fields and paragraphs that are allowed. In the relational model, repeating fields such as keys and authors are treated differently from non-repeating fields, such as the title. This is caused by the normalisation process where relations are designed so as to optimise updating. (For a more detailed discussion of normalisation problems see, for example, [2]). This creation of several relations from the original document necessitates the introduction of a new field which we have called D#. Its purpose is to relate together the entries in the different tables which were derived from the same original document. As we shall see, it is this normalisation process that causes many of the problems with the relational model. In the array model, the internal organisation parallels the external view of the document.

We will now look at some queries expressed in the notation of the three systems systems and compare them. There are a number of relational query languages, all with similar power. We will use SQL, probably the best known, to demonstrate the relational approach. To illustrate the array model, the query language we use is AQL. This is an experimental language currently under development. It combines the basic structure of SQL with a subset of the operations of NIAL, a programming language based on More´s work, [10]. More detailed descriptions of AQL are given elsewhere, [11,12].

In the following examples, the STAIRS query is given first followed by the same query expressed in SQL and AQL.

a) Find all documents dated "1984".

```
SELECT
   YEAR EQ 1984

SELECT D# FROM Document
   WHERE YEAR = 1984

SELECT FROM Documents
   WHERE Year = 1984
```

The first difference to note is that in SQL we specify what fields we want retrieved as part of the search command. In STAIRS this decision is postponed until the actual display or printing of the documents. In SQL, if we want to alter the displayed information, the search must be performed again. The AQL query is identical to the SQL query except for the minor difference that if the fields to be retrieved are not specified, the entire document is retrieved. More significantly, the array model does not prohibit indirection. In particular, it is possible to obtain a list of references (or pointers), to documents, rather than the actual documents themselves. For example, we can write:

```
SELECT REFERENCE FROM Documents
   WHERE Year = 1984
```

This type of retrieval more closely follows what happens in STAIRS.

In STAIRS, the results of the retrieval operation are numbered automatically and it is possible to refer to the results using this number in subsequent operations. In both SQL and AQL, the user can assign names to results as in:

```
   ASSIGN Myname: SELECT etc. (SQL)

   Myname GETS SELECT etc. (AQL)
```

b) Find any document whose title is "WAR AND PEACE".

In STAIRS we can do a search and then restrict the search against title paragraphs as follows:

```
   SEARCH
      1: "WAR" ADJ "AND" ADJ "PEACE"
      2: 1.TITLE

   SELECT D# FROM DOCUMENT
      WHERE Title = "WAR AND PEACE"

   SELECT FROM Documents
      WHERE Title = "War and Peace"
```

This query cannot really be specified in STAIRS without making "Title" into a formatted field. If it is an unformatted paragraph, the normal choice, the best we can do is search for the adjacency of the words of the title. In SQL and AQL, this query is structurally identical to the previous one. However in STAIRS we can search on the individual words of the title whereas in SQL and AQL it would first be necessary to build a separate structure to hold the words.

c) Find all documents containing the keyword "INFORMATION".

In STAIRS we would write:

```
   SEARCH
      KEYS.INFORMATION
```

In SQL, the corresponding query is:

```
   SELECT D# FROM Documents, Keys
      WHERE Key = "INFORMATION"
      AND Keys.D# = Documents.D#
```

In SQL, we have to relate information spread over two relations, the Document relation and the Keys relation. In STAIRS, the information is structured so that the Keys are integrated with the rest of the data in a single file structure. The same query in AQL is:

```
   SELECT FROM Documents
      WHERE Key = "INFORMATION"
```

It is obvious that even at this simple level, the SQL query is much more difficult to formulate than the other two.

It is interesting to look at the same query where we want to retrieve parts of the documents such as the title, authors and associated keys. In STAIRS there is no difference. We simply specify the fields of interest when we print. However in SQL the query becomes much more complex, since the required information is spread over three relations.

```
SELECT Title Author Key FROM Documents, Keys, Authors
   WHERE Key = "INFORMATION"

   AND Keys.D# = Documents.D#
   AND Authors.D# = Documents.D#
```

In AQL, the query remains largely the same:

```
SELECT Title (Authors) (Keys) FROM Documents
   WHERE Key = "INFORMATION"
```

Parenthesisation is used to denote non-atomic objects, or lists. This example illustrates the main problem with the relational model, namely normalisation. Not only is the complexity of the query increased, but the object retrieved is itself not a "natural" one. Say there was one original document which satisfied the query - "THE ARRAY MODEL", with keys "INFORMATION" and "DATABASE" and authors "MACLEOD" and "BARNARD". Then the retrieved object which the user would see would be:

THE ARRAY MODEL	INFORMATION	MACLEOD
THE ARRAY MODEL	DATABASE	MACLEOD
THE ARRAY MODEL	INFORMATION	BARNARD
THE ARRAY MODEL	DATABASE	BARNARD

This is not a very aesthetically pleasing representation of what is wanted. It can be contrasted with the same information as represented in the array model:

```
+---------------+------------------------+---------------------+
|THE ARRAY MODEL|+-----------+---------+ |+-------+-------+ |
|               ||INFORMATION|DATABASE| ||MACLEOD|BARNARD| |
|               |+-----------+---------+ |+-------+-------+ |
+---------------+------------------------+---------------------+
```

This is a much more natural representation in that it corresponds almost exactly to the original document.

d) Find all documents containing the keys "INFORMATION" and "ARRAY"

```
SEARCH
   1: INFORMATION AND ARRAY
   2: 1.KEYS

SELECT D# FROM Keys
   WHERE Key = "INFORMATION"
   INTERSECT
   ( SELECT D# FROM Keys
     WHERE Key = "ARRAY"
   )

SELECT FROM Documents
   WHERE ("INFORMATION" IN Keys)
     AND ("ARRAY" IN Keys)
```

Boolean searches are one of the most common retrieval strategies. The SQL formulation is extremely awkward and very inappropriate for iterative searching.

e) Find the titles of all documents for which Barnard is the first author.

In STAIRS we apparently cannot do this, despite the fact that the information is implicitly available in the file organisation. Nor, in the relational model, can we answer this query with the three relations we have currently defined. However, if the Authors relation was modified to contain an additional attribute, position, then the query could be processed.

```
SELECT Title FROM Documents, Authors
   WHERE Author = "Barnard"
   AND Author.Position = 1
   AND Authors.D# = Documents.D#
```

A major benefit of using a data base approach to document retrieval, is that the information organisation is not so rigorously pre-determined. It is relatively straightforward to add a new attribute to a relation or to add a new relation to the database. In STAIRS we have little flexibility in defining the structure of the database. On the other hand, it would be difficult to argue that the above query is either an elegant or natural formulation of the query being posed. However the equivalent AQL query is completely straightforward.

```
SELECT FROM Documents
   WHERE FIRST Authors = "BARNARD"
```

f) Find all titles with more than three authors.

Another interesting class of queries involves quantification. This cannot be done in STAIRS, (although again, the necessary information is available in the file organisation). In SQL we can write:

```
SELECT D# FROM Authors
    GROUP BY D#
    HAVING COUNT (*) > 3
```

It is, once again, fairly obvious from this example that SQL does not lend itself to the straightforward expression of many types of simple queries. Essentially, what this query does is sort the relation by D# and count all the tuples with the same D#. In effect, non-atomic objects are being temporarily constructed. In contrast, the AQL equivalent can be expressed quite naturally.

```
SELECT FROM Documents
    WHERE TALLY Authors > 3
```

THE NEED FOR A NEW MODEL

Our view is that information is not naturally viewed and managed as a collection of sets. Rather information has structure and this structure is part of the information content. When we flatten information we lose part of the information itself. Our example on position illustrates this point. In most relational systems it is probably impossible to find the first author of an article. It can be said say that if this is important, a new relation can be created or position could be defined as an attribute. This seems unsatisfactory. It means a conscious effort has to be made to represent structure through attributes. That is, we have to predict potential queries which seems to violate the relational ideology. A typical document is not solely composed of atomic objects but rather may contain lists of objects, such as multiple authors, keywords, references and so on. These lists have properties such as length and order. The relational model tends to become unwieldy when we try to represent such an object in a purely tabular form.

Probably the major way in which bibliographic data differs from conventional data base retrieval is that retrieval is non-deterministic. That is, two different users might well expect to obtain different results for the same initial query. Much of the user´s efforts in the bibliographic context is spent in interacting with the system in order to obtain the optimal group of documents related to the original query. In the conventional data base environment, the user typically spends more time in formulating the original query whereas in bibliographic data the query is continually reformulated or modified in response to the results obtained. Since continual viewing of the results is important in document retrieval, it is equally important that the structures implicit in the underlying data model reflect the structure of the information being retrieved. A basic problem with the relational model is that the internal representation can often be quite different from the external object. Furthermore, the intermediate object that must be constructed to rebuild the external object, is itself anomalous in that it is not optimally normalised. It is not natural to normalise data outside a computer information system and people don´t usually do it.

While the incorporation of text handling features into the relational model in order to enhance its suitability as an information retrieval system is obviously useful, this approach should not be regarded as the final solution of the general problem of integrating information retrieval and data base systems.

In some ways the array model is a generalisation of the relational model. In fact, by restricting arrays to tabular structures, a relational representation is possible and there are many occasions when tables are natural structures. However, it is our strong contention that tables are not sufficiently powerful to provide a universal structure for all information. Structure is important and if structured objects can be handled in a relatively straightforward non-procedural manner, then there is no reason why we should not develop information systems based on such structures.

The array model appears to provide a natural basis for this type of approach. It is noteworthy that the structure of an AQL query is very simple. In fact it is similar to the material being retrieved. Where such a natural sort of mapping between the query structure and the structure of the information being retrieved, it seems apparent that the underlying DBMS is an appropriate model.

REFERENCES

1. Macleod, I. A. "SEQUEL as a Language for Document Retrieval", _Journal of the American Society for Information Science_. Vol. 30, pp. 243-249, 1979.

2. CRAWFORD, R. G. "The Relational Model in Information Retrieval", in _Journal of the American Society for Information Science_. Vol. 32, No. 1 (January 1981), pp. 51-64.

3. Macleod, I. A. "A Data Base Management System for Document Retrieval Applications, _Information Systems_, Vol. 6, pp. 131-137, 1981.

4. Schek, H.J. and Pistor, P. "Data Structures for an Integrated Data Base Management and Information Retrieval System", _Proceedings of the Eighth International Conference on Very Large Data Bases_. Mexico City, pp.197-207, 1982.

5. Chamberlin, D. and Boyce, R. "SEQUEL: A Structured English Query Language", _Proceedings of the 1974 ACM-SIGMOD Workshop on Data Description, Access and Control_. Ann Arbor, Michigan, (May 1974), pp. 249-264.

6. IBM Corporation. Storage and Information Retrieval, (STAIRS) Reference Manual.

7. Bibliographic Retrieval Services Inc. BRS Bulletin, Schenectady, New York.

8. More, T. "A Theory of Arrays with Applications to Databases" Tech. Report G320-2106, IBM Scientific Centre, Cambridge, Mass., 1975.

9. More, T. "Notes on the Diagrams, Logic and Operations of Array Theory", Tech. Report G320-2137, IBM Scientific Centre, Cambridge, Mass., 1981.

10. Jenkins, M.A. "The Q'Nial Reference Manual, Tech. Report 82-123, Queen's University, Kingston, Ontario, 1982.

11. Macleod, I.A. "AQL - A Query Language for the Array Model", Technical Report, Department of Computing and Information Science, Queen's University, Kingston, Ontario, 1983.

12. Macleod, I.A. A Model for Integrated Information Systems, Proceedings of the 9th Conference on Very Large Data Bases, pp.280-289, Florence, 1983.