Division of Research
Graduate School of Business Administration
The University of Michigan

October 1984

AN EXTENDED RELATIONAL DOCUMENT RETRIEVAL MODEL

Working Paper No. 394

David C. Blair

The flexibility of the relational logical model in data base system design is well known, so the application of relational design structures to non-data retrieval situations is not surprising. A recent spate of papers applying relational logical structures and query languages to document, or information, retrieval systems [Crawford, Macleod, 1979, 1981] demonstrate the broad applicability of these techniques. These are not entirely new ideas. The first information retrieval system to use a relational logical structure, the Relational Data File, [Levien and Maron, 1965, 1966] was primarily a document retrieval system, and a 1973 paper showed the efficacy of the early relational data manipulation language SQUARE for querying document data bases [Blair, 1974].

The work to date in relational models of document retrieval has been largely preliminary and has modeled only relatively simple document retrieval situations. This paper will describe an extended relational model for document retrieval and will discuss some retrieval considerations of particular importance for document retrieval.

Document Retrieval: The Central Problem

One of the principal advantages which relational logical structures offer over the earlier logical structures, hierarchical and net work (CODASYL), is the comparative ease with which an inquirer using a relational data base can construct ad hoc queries--queries which are not routinely or repeatedly asked of the system. The capacity to be able to answer ad hoc inquiries easily is an advantage in data base management systems, but not usually a necessity. For document retrieval, on the other hand, the ability to answer ad hoc inquiries is a necessity. [Blair, 1984]. Document retrieval systems have been implemented using

logical data base structures and query languages which preceded relational designs [Dattola], but such systems are comparatively difficult to use in ad hoc inquiry and, resultingly, will be difficult to use to implement all but the simplest logical models of document retrieval.

Ad hoc inquiry is important for document retrieval systems because of the tremendous variety in the way that people search for needed documents. Inquirers may want documents because they are authored by particular individuals, are published during a particular time frame, appear in one or several journals, concern a particular subject, are written by authors who are affiliated with certain institutions, are of a particular type (e.g., article, letter, report, conference proceedings, etc.), or any complex combination of such search categories. Document retrieval systems which do not permit this kind of search flexibility greatly reduce the chances of an inquirer's retrieving useful articles. But the formulation of a wide variety of ad hoc queries is not the only important capability of an advanced document retrieval system. Such a system should enable the inquirer to retrieve not just information about individual documents but also information about the aggregate of documents ("meta-information", if you will). Information such as, for example, "what subject headings are most frequently applied to documents authored by individuals working at institute X". This is not specifically a request for documents, but a request for important tacit information that may be derivable from the document descriptions in the data base. The ability to retrieve such tacit information is an important capability of an advanced document retrieval system. The following discussion will show how such

tacit information may be retrieved through the use of relational logical structures and Data Manipulation Languages.

Basic Relational Structure

The basic (normalized) relational structure of the initial document retrieval facility would look like this:

Relation Name	Attributes
CITATION	DOCUMENT #, TITLE, DOCUMENT TYPE, PUBLICATION DATE, JOURNAL NAME, VOLUME, NUMBER, PAGES
ABSTRACT	DOCUMENT #, ABSTRACT
AUTHOR	DOCUMENT #, NAME
DIRECTORY	NAME (author's name), INSTITUTION
INSTITUTE	INSTITUTION (name), TYPE, ADDRESS, PHONE
JOURNAL	JOURNAL NAME, PUBLISHER

Such a set of relations would represent a basic document retrieval schema, which could be enhanced in several ways. But before we look at the enhancements we should look at some of the inquiries which the basic logical structure can support. For readability, I will use the SQL query language [Date] when describing how actual queries would be constructed, since SQL is relatively understandable even to those with no familiarity with relational query languages.

Typical Queries

1. What are the titles of articles written by Raymond Larsen?

```
SELECT TITLE
FROM   CITATION
WHERE  DOCUMENT # =
       SELECT DOCUMENT #
       FROM   AUTHOR
       WHERE  NAME = 'Raymond Larsen'
```

2. Retrieve the abstracts of the articles written by Raymond Larsen.

```
SELECT ABSTRACT
FROM   ABSTRACT
WHERE  DOCUMENT # =
       SELECT DOCUMENT #
       FROM   AUTHOR
       WHERE  NAME = 'Raymond Larsen'
```

3. Where does Raymond Larsen work?

```
SELECT INSTITUTION
FROM   DIRECTORY
WHERE  NAME = 'Raymond Larsen'
```

4. What is Raymond Larsen's address?

```
SELECT ADDRESS
FROM   INSTITUTE
WHERE  INSTITUTION =
       SELECT INSTITUTION
       FROM   DIRECTORY
       WHERE  NAME = 'Raymond Larsen'
```

5. Which authors of articles in our data base are affiliated with the University of California?

```
SELECT NAME
FROM   DIRECTORY
WHERE  INSTITUTION = 'University of California'
```

A frequent service provided by many information centers is to send individuals a list of the contents of specified journals or magazines. Such a service could be easily provided in the sample data base model:

6. What are the titles of the articles appearing in April 8 issue of Communications of the ACM?

```
SELECT TITLE
FROM   CITATION
WHERE  JOURNAL NAME = 'Communications of the ACM'
AND    PUBLICATION DATE = 03/08/83
```

This request could be made by specifying the VOLUME and NUMBER of the journal if that information is more readily available than the date. Sometimes, though, the inquirer may not even know the date or volume number of the journal whose contents he wants. He may only want to know the contents

of the "most recent" copy of the desired journal which exists on the data base. Such a request can be accommodated by using an arithmetic function which exists in all major relational data base management systems:

7. What are the titles of the articles appearing in the most recently received issue of Communications of the ACM?

```
SELECT TITLE
FROM   CITATION
WHERE  JOURNAL NAME = 'Communications of the ACM'
AND    PUBLICATION DATE = MAX
```

Conjunctive queries such as the two above are known in the document retrieval vernacular as "Boolean Queries". Such queries are quite frequent in document retrieval.

Another type of "current contents" request occurs in response to the inquirer who wants to see the titles of articles appearing in a specific journal (or journals) over a period of time. Such an information need may occur when an individual is doing a literature search for certain types of articles, is catching up on his reading in a journal he has not seen in a while, or is looking for a specific article whose title he can recognize but whose date of publication he has forgotten:

8. Give me the titles of all articles published in Communication of the ACM since 1980.

```
SELECT TITLE
FROM   CITATION
WHERE  JOURNAL = 'Communications of the ACM'
AND    PUBLICATION DATE > 12/31/79
ORDER BY PUBLICATION DATE DESCENDING
```

The final statement in the above request insured that the titles will be in chronological order with the most recent first.

For individuals who want to be informed regularly about the content of journals which are added to the document data base, a complex Boolean request could be kept on file for that individual which includes the names

of all the journals (and any other document parameters) of which he wants to be informed. The request would merely be run at periodic intervals and the results forwarded to the requesting individual. To insure that only the contents of journals added since the inquirer's last request are returned, the request should be made using the Acquisition Date rather than the Publication Date as a search parameter. For example:

9. Give me the titles of all the articles published in Communications of the ACM and Computer Journal since my last request (1/31/83).

```
SELECT TITLE
FROM   CITATION
WHERE  JOURNAL IN ('Communications of the ACM', 'Computer Journal')
AND    ACQUISITION DATE > 1/31/83
```

Searching by acquisition date has three advantages: (1) The inquirer is not overloaded with previously retrieved material; (2) The request can be processed very efficiently since only a small part of the data base (those articles added since 1/31/83) needs to be accessed; (3) It will retrieve older journal articles which have only recently been added to the data base (such as, in our example, a back issue of Computer Journal). This is not possible if retrieval is based on Publication Date.

THE EXTENDED RELATIONAL MODEL

By adding several relations to the schema described on page 1 we can greatly increase the kinds of queries which can be answered by the retrieval system. The most important addition to the data base schema at this point would be to introduce some kind of subject-access capability. This can be done through the following relation:

KEYWORDS DOCUMENT#, KEYWORD, WEIGHT

This relation relates subject descriptions (keywords to specific documents. It also includes a weight (usually between 0 and 1.0) which reflects

how applicable a particular subject description is to a given document. For example, a document might be represented as follows:

```
#4357  Special Computers  0.5
#4357  Programming        0.7
#4357  File Organization  0.9
         .
         .
         .
```

This would indicate that document #4357 deals with programming and file organization on special computers, and that it is largely about file organization. With this kind of structure there is no limit to the number of keywords which could be assigned to a particular document, but if we look at existing systems we see that usually 6-10 keywords are typically used to describe the subject content of a document.

Now we can retrieve documents from the system based on subject specifications:

10. Retrieve the documents which concern "occupational retaining" and have an assigned subject weight of greater than 0.7.

```
SELECT *
FROM   CITATION
WHERE  DOCUMENT # =
       SELECT DOCUMENT #
       FROM   KEYWORDS
       WHERE  KEYWORD = 'occupational retraining'
       AND    WEIGHT > 0.7
       ORDER BY WEIGHT DESCENDING
```

(The "*" in the SELECT statement indicates that the entire tuple is to be retrieved.)

It is very useful for a document retrieval system to be able to order retrieved citations according to some priority (such as, above, where they are ordered by descending keyword weight). One of the most persistent problems in document retrieval is "ouput overload"--the condition where too many documents (i.e., document citations) are retrieved for the inquirer

to browse through to find the documents he/she wants [Blair, 1980]. In many modern computerized retrieval systems, a request like number 10 with a single keyword specification may retrieve thousands (or, even tens of thousands) of document citations. All of the retrieved citations will have had, by definition, the specified keyword assigned to them, but many of the citations refer to documents which are only marginally concerned with the subject indicated by the keyword. By ordering the retrieved citations according to keyword weight, the inquirer insures that even where large numbers of citations are retrieved, the ones that more closely match the request are presented to him/her first. Consequently, even retrieved sets of thousands of documents may not be an impediment to effective retrieval since the citations which are more likely to satisfy the inquirer will be ranked first. The "ORDER BY" and "GROUP BY" commands in SQL give the inquirer wide latitude in prioritizing output.

Because of the sociological nature of research and industry, the name of an individual who works in a relevant area of research can be used as a clue to find other relevant information on the data base, by using the KEYWORDS relation in combination with the other relations to retrieve relevant documents. For example:

11. Give me the titles of articles on "expert systems" which are published by individuals who are affiliated with the same institute with which John Murphy is affiliated.

```
SELECT TITLE
FROM   CITATION FIRST
WHERE  FIRST.DOCUMENT # = ANY
       SELECT DOCUMENT #
       FROM CITATION SECOND, KEYWORDS
       WHERE KEYWORDS.KEYWORD = "expert systems"
       AND   SECOND. CITATION NAME = ANY
             SELECT NAME
             FROM   DIRECTORY FIRST
             WHERE  FIRST.INSTITUTION =
                    SELECT INSTITUTION
                    FROM   DIRECTORY SECOND
                    WHERE  SECOND.NAME = "Murphy, John"
```

(The "WHERE FIRST.DOCUMENT # = ANY" command indicates that titles should be retrieved whose document #s are "any" of the ones found by the following SELECT commands. The CITATION FIRST and CITATION SECOND specification is a SEQUEL convention which allows the results of one search of the CITATION relation to be used as arguments for a second search of the CITATION relation.)

Sometimes a subject search must be "reversed" for those inquirers who are familiar with the literature of the data base but are not familiar with the exact subject descriptions being used (even a minor spelling error in keyword specification might lead to poor retrieval results). To "get into" the system an inquirer might retrieve the keywords which are used to index a document in which he is interested and which is already on the data base. He would then use those retrieved keywords to formulate a conventional subject request to the system to retrieve other documents on the same subject.

12. Give me the keywords used to describe the document "Process control in shop-floor automation" by Molly Bloom.

```
SELECT KEYWORD
FROM   KEYWORDS
WHERE  DOCUMENT # =
       SELECT DOCUMENT #
       FROM   CITATION, AUTHOR
       WHERE  CITATION.DOCUMENT # = AUTHOR.DOCUMENT #
       AND    CITATION.TITLE = 'Process control in shop-floor
                               automation'
       AND    AUTHOR.NAME = 'Molly Bloom'
```

This search could be combined into one query as follows:

13.
```
SELECT TITLE
FROM   CITATION
WHERE  DOCUMENT # =
       SELECT FIRST.DOCUMENT #
       FROM   KEYWORDS FIRST
       WHERE  FIRST.WEIGHT > 0.5
       AND    FIRST.KEYWORD = ANY
              SELECT SECOND.KEYWORD
              FROM   KEYWORDS SECOND
              WHERE  SECOND.DOCUMENT # =
                     SELECT DOCUMENT #
                     FROM   CITATION, AUTHOR
                     WHERE  CITATION.DOCUMENT # = AUTHOR.DOC #
                     AND    CITATION.TITLE = 'Process control in
                                             shop-floor automation'
                     AND    AUTHOR.NAME = 'Molly Bloom'
```

The principal disadvantage of query 13 is that the inquirer loses some of the query-formulation control which he would have if he conducted the search as a two-stage process.

INFERENTIAL RETRIEVAL IN RELATIONAL DATA BASES

A relational document retrieval system does not just contain documents, it also contains a great deal of valuable information of an inferential or tacit nature. For example, an inquirer may want to know what the major

journal sources are in the field of "flexible manufacturing" so he will be certain to keep up to date on their articles:

```
14.  SELECT UNIQUE JOURNAL NAME, COUNT(DOCUMENT #)
     FROM   CITATION
     WHERE  DOCUMENT # = ANY
            SELECT DOCUMENT #
            FROM   KEYWORDS
            WHERE  KEYWORD = 'flexible manufacturing'
            ORDER BY COUNT(DOCUMENT #) DESCENDING
```

The command "COUNT(DOCUMENT #)" keeps a running total of the number of documents which have appeared in a given journal and have been assigned the subject description "flexible manufacturing". The "ORDER BY . . ." command insures that the output will consist of a rank ordering of journal titles arranged in descending order by how many articles on "flexible manufacturing" have appeared in them. Often, an inquirer can infer a lot about what kind of research may go on at a particular institution just by looking at the kinds of publications, memos, or reports which are produced by individuals affiliated with that institution. By tabulating the information in certain ways, some interesting relationships may be revealed. For example:

15. Rank the institutions by how many authors they have who publish in 'flexible manufacturing'.

```
SELECT UNIQUE INSTITUTION, COUNT(UNIQUE NAME)
FROM   DIRECTORY
WHERE  NAME = ANY
       (SELECT NAME
        FROM   AUTHOR
        WHERE  DOCUMENT # = ANY
               SELECT DOCUMENT #
               FROM   KEYWORDS
               WHERE  KEYWORD = 'flexible manufacturing')
               ORDER BY COUNT(UNIQUE NAME) DESCENDING
```

The results of the above search could be compared to the total number of authors at each institution to get an idea of the percentage of concentration that an institution has in 'flexible manufacturing'.

The subject terms which have been assigned to documents in the data base can also be used to derive a rough "subject profile" of the research at a particular institution by asking the following query:

16. List the different keywords which have been assigned to articles produced by individuals affiliated with the General Motors Institute, and count the number of documents to which each of these keywords have been assigned.

```
SELECT UNIQUE KEYWORD, COUNT(DOCUMENT #)
FROM   KEYWORDS
WHERE  DOCUMENT # = ANY
       (SELECT DOCUMENT #
        FROM   AUTHOR
        WHERE  NAME = ANY
               SELECT NAME
               FROM   DIRECTORY
               WHERE  INSTITUTION = 'General Motors Inst.')
               ORDER BY COUNT (DOCUMENT #) DESCENDING
```

Occasionally, it may be important to use the information on the data base to generate a list of institutions who might be interested in receiving information on a particular area. This might be done with the following query:

17. Get the names and addresses of all research groups who have at least one member who has published a recent (1981 or after) paper on 'integrated manufacturing'.

```
SELECT INSTITUTION, ADDRESS
FROM   INSTITUTE
WHERE  INSTITUTION =
       SELECT INSTITUTION
       FROM   DIRECTORY
       WHERE  NAME =
              SELECT NAME
              FROM   CITATION
              WHERE  DATE > 12/31/80
              AND    NAME =
                     SELECT NAME
                     FROM   AUTHOR
                     WHERE  DOCUMENT # =
                            SELECT DOCUMENT #
                            FROM   KEYWORDS
                            WHERE  KEYWORD = 'integrating manu-
                                             facturing'
```

ASSOCIATIVE SEARCHING USING THE RELATIONAL MODEL

One of the most important facilities of a good document retrieval system is its associative searching capability. This permits the inquirer to discover semantic relationships between the subject index terms which have been assigned to documents on the data base. One of the simplest and most useful statistics for infering semantic relationships between subject terms (keywords) is the percentage of co-occurrence of assignment of these terms. This percentage expresses a probability that if keyword "X" is assigned to a particular document, then there is a calculatable probability that keyword "Y" will also be assinged to that document. This probability is merely the percentage of times that keyword "Y" has been assigned to documents which have keyword "X" assigned. (Note that the probability of "Y" being assigned given the assignment of "X" is not the same as the probability of "X" being assigned given the assignment of "Y".) The primary use of associative searching is to semantically "broaden" an inquirer's subject search. For example, if an inquirer exhausts his search for documents with the keyword 'flexible manufacturing' he can retrieve a list of co-occurring subject terms by using the following relation:

THESAURUS KEYWORD, COOCCURRING TERM, PERCENT

A typical query might be:

18. Retrieve the keywords which co-occur with the keyword "Air pollution' which have a probability of co-occurrence greater than .040. Rank these terms by decreasing probability of co-occurrence.

```
SELECT COOCCURRING TERM, PERCENT
FROM   THESAURUS
WHERE  KEYWORD = 'Air pollution'
AND    PERCENT > .040
ORDER BY PERCENT DESCENDING
```

The output of such a search might look something like:

KEYWORD	COOCCURRING TERM	PERCENT
Air Pollution	Dust	.479
	Waste Disposal	.384
	Water Supply	.231
	Quarrying	.132
	Noise	.132
	Poison	.126
	Environment	.101
	Pesticide	.089
	Occupational Safety	.081
	Gas Industry	.063
	Chemical Industry	.061
	Education	.057
	Natural Resources	.045

Such a list has two principal uses: (1) It can, as mentioned before, be used by inquirers who want to find semantically related keywords (the assumption being that keywords which have a high probability of co-occurring are semantically related). (2) It can be used by indexers to assist them in the indexing process (an indexer would only have to identify the keyword which identifies the main subject of the document to be indexed, and could then select the appropriate secondary subject categories from the list of keywords which co-occur with the principal keyword).

An inquirer could expand his or her search without consulting a co-occurrence list by entering the following formal query:

```
19.  SELECT TITLE
     FROM   CITATION
     WHERE  DOCUMENT # = ANY
            SELECT DOCUMENT #
            FROM   KEYWORDS FIRST, KEYWORDS SECOND
            WHERE  FIRST.KEYWORD = 'shop automation'
            AND    SECOND.KEYWORD = 'integrated manufacturing'
            AND    FIRST.DOCUMENT # = SECOND.DOCUMENT #
```

A simple disjunctive query could be handled as follows:

21. Retrieve the titles of all the documents which have been indexed with either keywords 'shop automation' or 'integrated manufacturing'.

```
SELECT TITLE
FROM   CITATION
WHERE  DOCUMENT = ANY
       SELECT DOCUMENT #
       FROM   KEYWORDS
       WHERE  KEYWORDS IN
              ('shop automation', 'integrated manufacturing')
```

Subject searching is a non-deterministic process in which several topic alternatives often must be described in an inquirer's query. These alternatives are represented by complex conjunctive and disjunctive Boolean combinations of keywords. For example:

22. Retrieve the titles of all the documents which are indexed with either 'shop automation', 'computerization', or 'automation', and either 'integrated manufacturing' or 'flexible manufacturing'. (This query is the conjunction of two disjunctive sets of three keywords and two keywords, respectively)

```
SELECT TITLE
FROM   CITATION
WHERE  DOCUMENT # = ANY
       SELECT DOCUMENT #
       FROM   KEYWORDS FIRST, KEYWORDS SECOND
       WHERE  FIRST.DOCUMENT # = SECOND.DOCUMENT #
       AND    FIRST.KEYWORD IN ('shop automation', 'computerization',
              'automation')
       AND    SECOND.KEYWORD IN ('integrated manufacturing', 'flex-
              ible manufacturing')
```

A general formulation is possible for constructing Boolean subject queries, providing a format for even the most complex keyword queries:

```
SELECT FIRST.DOCUMENT #
FROM   KEYWORDS FIRST
       KEYWORDS SECOND
            .
            .
            .
       KEYWORDS Nth
WHERE  FIRST.DOCUMENT # = SECOND.DOCUMENT #
AND    SECOND.DOCUMENT # = THIRD.DOCUMENT #
 .
 .
 .
AND    N-1.DOCUMENT # = Nth.DOCUMENT #
AND    FIRST.KEYWORD IN ('xxxx',..., 'xxxx')
AND    SECOND.KEYWORD IN ('xxxx',..., 'xxxx')
 .
 .
 .
AND    Nth.KEYWORD IN ('xxxx',..., 'xxxx')
```

The logical format of this kind of query can be represented in the propositional calculus as follows:

$$(K_{a_1} \text{ v } K_{b_1} \text{ v } \ldots \text{v } K_{n_1}) \cdot (K_{a_2} \text{ v } K_{b_2} \ldots \text{v } K_{n_2}) \cdot \ldots \cdot$$

$$(K_{a_r} \text{ v } K_{b_r} \text{ v } \ldots \text{v } K_{n_r})$$

[where the symbols "v" and "·" represent disjunction and conjunction, respectively, and "K" stands for a keyword.]

This particular logical pattern is, of course, conjunctive normal form, and while many Boolean expressions are not in conjunctive normal form, they all can be transformed into conjunctive normal form without loss of meaning or well-formedness. This means that any Boolean retrieval query can be represented in the above format. For example:

$$(K_a \cdot K_b) \text{ v } K_c$$

which is not in conjunctive normal form, can be represented by the equivalent logical construct:

$$(K_a \text{ v } K_c) \cdot (K_b \text{ v } K_c)$$

Or, in another example, the Boolean query:

$$(K_a \cdot K_b) \text{ v } (K_c \cdot K_d)$$

can be represented by the equivalent conjunctive normal form expression:

$$(K_a \text{ v } K_c) \cdot (K_a \text{ v } K_d) \cdot (K_b \text{ v } K_c) \cdot (K_b \text{ v } K_d)$$

Those readers familiar with propositional logic will, no doubt, have observed that the expression "$(K_A \cdot K_B) \text{ v } (K_C \cdot K_D)$", while not in <u>conjunctive</u> normal form, is in <u>disjunctive</u> normal form. Since all Boolean expressions can be non-loss transformed into either conjunctive or disjunctive normal form, it appears that the recommendation to convert all complex Boolean SQL

queries into conjunctive normal form is somewhat arbitrary. This is not the case. Conjunctive normal form expressions are more easily represented in SQL than disjunctive normal form expressions. The general SQL format for disjunctive normal form expressions looks like:

```
SELECT UNIQUE FIRST.DOCUMENT
FROM   KEYWORDS FIRST
WHERE  FIRST.DOCUMENT # = ANY
       (SELECT  N1.DOCUMENT #
       FROM     KEYWORDS N1
                KEYWORDS (N1+1)
                   •
                   •
                   •
                KEYWORDS (N1+M1)
       WHERE    N1.DOCUMENT # = (N1+1).DOCUMENT #
       AND      (N1+1).DOCUMENT # = (N1+2).DOCUMENT #
        •
        •
        •
       AND      (N1+M1-1).DOCUMENT # = (N1+M1).DOCUMENT #
       AND      N1.KEYWORD = "KA1 "
       AND      (N1+1).KEYWORD = "KB1 "
        •
        •
        •
       AND      (N + M).KEYWORD = "KN1 "
OR
       SELECT   N2.DOCUMENT #
       FROM     KEYWORDS.N2
                KEYWORDS.(N2+1)
                   •
                   •
                   •
                KEYWORDS.(N2+M2)
       WHERE    N2.DOCUMENT # = (N2+1).DOCUMENT #
       AND      (N2+1).DOCUMENT # = (N2+2).DOCUMENT #
        •
        •
        •
       AND      (N1+M -1).DOCUMENT # = (N2+M2).DOCUMENT #
       AND      (N2+1).KEYWORD = "KA2 "
       AND      (N2+1).KEYWORD = "KB2 "
        •
        •
        •
       AND      (N +M ).KEYWORD = "KN2 "
```

```
OR
    •
    •
    •
OR
    SELECT    N_n.DOCUMENT #
    FROM      KEYWORDS.N_n
              KEYWORDS.(N_n+1)
                 •
                 •
                 •
     WHERE    N_n.DOCUMENT # = (N_n+1).DOCUMENT #
     AND      (N_n+1).DOCUMENT # = (N_n+2).DOCUMENT #
      •
      •
      •
     AND      (N_n+M_n-1).DOCUMENT # = (N_n+M_n).DOCUMENT #
     AND      N_n.KEYWORD = "K_An "
     AND      (N_1+1).KEYWORD = "K_Bn "
      •
      •
      •
     AND      (N_n+ M_n).KEYWORD = "K_Nn ")
```

This is clearly a more complex query format than the one for conjunctive normal form (q.v.). The minor inconvenience of converting an expression from disjunctive normal form to conjunctive normal form would be more than offset by the comparative ease of transforming conjunctive normal form expressions into SQL (or any other relationally complete DMLs) commands.

From this discussion of SQL query formulation we can see that while SQL is a "friendly" language, some Boolean queries may be translated into SQL commands only with great difficulty. To facilitate query construction complex Boolean queries should be reduced to their simplest form before they are translated into SQL commands. For example, the laws of propositional logic enable us to reduce the Boolean expression:

$$(K_p \cdot K_q) \vee (K_q \cdot K_r) \vee (K_p \cdot K_r) \vee K_r$$

to:

$(K_p \cdot K_q) \vee K_r$ [vid. Appendix A]

This, in turn, is translatable into the conjunctive normal form expression:

$(K_p \vee K_r) \cdot (K_q \vee K_r)$

This is a much easier expression to translate into SQL than the original one.

STORAGE STRUCTURE CONSIDERATIONS IN THE DOCUMENT RETRIEVAL MODEL

One of the problems with data bases which contain documents (as opposed to data bases which contain only data) is that because of the many different attributes of documents (e.g., author, title, data, journal, keywords, etc.) one document may be represented by 10-15 different tuples. This means that the size of the document data base will increase quite dramatically as documents are added to the collection (though this increase may be slowed by the careful use of data compression facilities and the judicious linking of the physical instantiations of the document attributes). If we assume the above logical structure and a mean subject indexing depth of 6 keywords, then the addition of one document to the collection results in a (theoretical) addition of at least 12 tuples to the data base (the number of tuples which must be added to the physical instantiation of the THESAURUS relation depends on how many of the new document's keywords are new to the vocabulary of the data base). This rapid growth of document data base size can cause problems both in the physical storage of attribute data and in the searching of the data base by inquirers.

The largest relation in the data base will be the KEYWORDS relation which needs a tuple for each assignment of a subject term to a document. With a data base of 10,000 documents and a mean indexing depth of 6, we would expect to have a vocabulary of 3,400-7,000 unique subject terms, and

a total number of indexing assignments of 59,880 (vid. Appendix B). The latter figure will be the number of tuples in the KEYWORDS relation.

The total number of tuples in the THESAURUS relation is equal to the number of unique co-occurrences of indexing terms in the data base (i.e., the number of distinct pairs of terms which appear together indexing a particular document). Given a data base of 10,000 documents and a mean indexing depth of 6, the estimated number of tuples for the THESAURUS relation would be 32,600 (vid. Appendix C).

In aggregate, then, the total number of tuples needed to build a data base of 10,000 documents is estimated to be 129,480 (vid. Table 1). Clearly, to implement a document retrieval system on a relational data base requires a substantial commitment of available resources. This may not be a problem on a data base management system running on a large mainframe computer or a smaller computer with a backend data base machine (such as, a VAX with Britten Lee's IDM 600), but to implement a working document retrieval system on a smaller computer (perhaps even a micro-computer) and and still maintain the same retrieval capabilities will require some changes in our basic model.

Relation	Tuples
CITATION	10,000
ABSTRACT	10,000
AUTHOR	10,000[1]
DIRECTORY	5,000[2]
INSTITUTE	1,000[3]
SOURCE	1,000[4]
KEYWORDS	59,880[5]
THESAURUS	32,600
TOTAL	129,480

1. Assumes only single-author documents.

2. Assumes only 5,000 unique authors in the data base.

3. Assumes many authors will be affiliated with the same institution.
4. Assumes many documents will be published in the same journal.
5. Tuples in the KEYWORDS relation will be equal to the total number of index term assignments in the data base.

Table 1

REDUCING STORAGE STRUCTURE

One of the observed characteristics of document retrieval systems is that retrieval patterns often follow a Pareto Distribution. [..] That is, about 20% of the documents on the data base will account for approximately 80% of the retrieval activity. In other words, a small "core" of documents will be retrieved repeatedly. Since the THESAURUS relation contains information about the statistical (and, by inference, semantic) relationships between assigned index terms, these relationships may be accurately modeled by using co-occurrence data from just the "core" documents rather than all the documents on the data base. The core documents can be easily identified by maintaining a count of the number of times each document on the data base is retrieved. The core documents are those which have been retrieved, or retrieved a number of times above an established cut-off value. In our example, if we assume that the core documents represent 20% of the data base, then the THESAURUS relation can be constructed on data from 2,000 documents rather than 10,000. Using the same methods which we used above, we find that we would only need an estimated 1,500 index terms to describe the subject content of these core documents, assuming a mean indexing depth of 6 (vid. Appendix B). The approximate number of unique

co-occurrences which are likely to occur for 1,500 terms and 2,000 documents is 7,550 (vid. Appendix C). This is the number of tuples needed to build a THESAURUS relation using data from only the core documents. The total number of tuples estimated to exist in the reduced data base is 104,430--a reduction in storage space of 19%.

If greater reductions in storage space are required it would probably not be wise to base the THESAURUS construction on a subset of the data base smaller than the set of core documents. Further reductions in storage structure size can be effected by reducing the mean indexing depth of keyword indexing (although, naturally, this may not be an easy or desirable policy to implement). If we were able to reduce indexing depth from 6 to 4 then the following changes would occur: 1. The KEYWORDS relation would be reduced from, approximately, 59,880 tuples to 39,824. 2. The THESAURUS relation would be further reduced from the core document level of 7,550 tuples to 3,570. In aggregate, the data base would now contain an estimated 80,394 tuples as compared with 104,430 tuples (core documents only, mean depth of 6) or with the original data base size of 129,480 tuples (all documents, means depth of 6). This would represent a 38% reduction from the full data base size.

The notion of a core of comparatively highly retrieved documents can also be a useful tool for inquirers. We can add a RETRIEVAL relation to the data base defined as follows:

RETRIEVAL DOCUMENT # TIMES (retrieved)

An inquirer would greatly speed up his search by limiting his requests for documents to those documents which have been retrieved one or more times, e.g.,

```
SELECT . . .
FROM   . . .
WHERE  . . .
       .
       .
       .
       AND DOCUMENT # = ANY
       SELECT DOCUMENT #
       FROM RETRIEVAL
       WHERE TIMES > 0
```

This would insure that the inquirer would see the more highly retrieved (and, by inference, more useful) documents first. This would mitigate the problem of output overload, which we discussed before. If the inquirer did not find all the documents he wanted, he could then expand his search to the rest of the data base by dropping the final SELECT clause.

The RETRIEVAL relation could also be used as the basis for ranking output. The inquirer would merely include a command in his document request to rank the output by the number of times the documents have been retrieved. This assumes that the more highly retrieved documents are more likely to be useful to the inquirer.

Conclusion

Recent research has shown that data base management systems are effective tools for constructing operational document retrieval systems. This discussion has argued that the retrieval requirements of document retrieval systems can be supported most effectively by the relational model, especially if a system capable of more advanced document retrieval techniques, such as associative or inferential retrieval, is desired. The logical structure for implementing the advanced or extended document retrieval model was discussed at length, and several storage structure issues have been addressed which are of particular importance for the design of document retrieval systems.

Appendix A

1. $(K_p \cdot K_q) \vee (K_q \cdot K_r) \vee (K_p \cdot K_r) \vee K_r$
2. $(K_p \cdot K_q) \vee (K_q \cdot K_r) \vee (K_p \cdot K_r) \vee (K_r \cdot 1)$ [identity]
3. $(K_p \cdot K_q) \vee (K_q \cdot K_r) \vee [(K_p \vee 1) \cdot K_r)]$ [distribution]
4. $(K_p \cdot K_q) \vee (K_q \cdot K_r) \vee [(1 \cdot K_r)]$ [identity]
5. $(K_p \cdot K_q) \vee (K_q \cdot K_r) \vee K_r$ [identity]

[repeat steps 2-5]

.
.
.

9. $(K_p \cdot K_q) \vee K_r$
10. $(K_p \vee K_r) \cdot (K_q \vee K_r)$ [distribution]

Another example:

$F = [(K_b \cdot \overline{K}_c) \vee (\overline{K}_b \cdot K_c)] \cdot [(\overline{K}_a \cdot K_b) \vee (\overline{K}_a \cdot K_c) \vee (K_a \cdot \overline{K}_b \cdot \overline{K}_c)]$

Set $F_1 = (K_b \cdot \overline{K}_c) \vee (\overline{K}_b \cdot K_c)$

and, $F_2 = (\overline{K}_a \cdot K_b) \vee (\overline{K}_a \cdot K_c) \vee (K_a \cdot \overline{K}_b \cdot \overline{K}_c)$

1. $F_1 = (K_a \cdot K_b \cdot \overline{K}_c) \vee (\overline{K}_a \cdot K_b \cdot \overline{K}_c) \vee (K_a \cdot \overline{K}_b \cdot K_c) \vee (\overline{K}_a \cdot \overline{K}_b \cdot K_c)$

   [change to complete disjunctive normal form]

2. $F_1' = (\overline{K}_a \cdot K_b \cdot K_c) \vee (\overline{K}_a \cdot \overline{K}_b \cdot \overline{K}_c) \vee (K_a \cdot K_b \cdot K_c) \vee (K_a \cdot \overline{K}_b \cdot \overline{K}_c)$

   [complement of F_1]

3. $F_1 = (F_1')' = (K_a \vee \overline{K}_b \vee \overline{K}_c) \cdot (K_a \vee K_b \vee K_c) \cdot (\overline{K}_a \vee \overline{K}_b \vee \overline{K}_c) \cdot (\overline{K}_a \vee K_b \vee K_c)$

   [complete conjunctive NF of F_1]

4. $F_2 = (\overline{K}_a \cdot K_b \cdot K_c) \vee (\overline{K}_a \cdot K_b \cdot \overline{K}_c) \vee (\overline{K}_a \cdot \overline{K}_b \cdot K_c) \vee (K_a \cdot \overline{K}_b \cdot \overline{K}_c)$

   [change to complete disjunctive NF]

5. $F_2' = (\overline{K}_a \cdot \overline{K}_b \cdot \overline{K}_c) \vee (K_a \cdot K_b \cdot K_c) \vee (K_a \cdot \overline{K}_b \cdot K_c) \vee (K_a \cdot K_b \cdot \overline{K}_c)$

   [complement of F_2]

6. $F_2 = (F_2')' = (K_a \vee K_b \vee K_c) \cdot (\overline{K}_a \vee \overline{K}_b \vee \overline{K}_c) \cdot (\overline{K}_a \vee K_b \vee \overline{K}_c) \cdot (\overline{K}_a \vee \overline{K}_b \vee K_c)$

   [complete conjunctive NF of F_2]

7. $F = F_1 \cdot F_2$

$F = (K_a \vee K_b \vee K_c) \cdot (K_a \vee \overline{K}_b \vee \overline{K}_c) \cdot (\overline{K}_a \vee K_b \vee K_c) \cdot (\overline{K}_a \vee K_b \vee \overline{K}_c)$

$\cdot (\overline{K}_a \vee \overline{K}_b \vee K_c) \cdot (\overline{K}_a \vee \overline{K}_b \vee \overline{K}_c)$ [conjunction of 3 and 6]

8. $F' = (K_a \vee \overline{K}_b \vee K_c) \cdot (K_a \vee K_b \vee \overline{K}_c)$

[complement of F]

9. $F = (F')' = (\overline{\overline{K}}_a \cdot K_b \cdot \overline{\overline{K}}_c) \vee (\overline{\overline{K}}_a \cdot \overline{\overline{K}}_b \cdot K_c)$

[complete disjunctive NF of F]

10. $\overline{K}_a \cdot [(K_b \cdot \overline{K}_c) \vee (\overline{K}_b \cdot K_c)]$

11. $\overline{K}_a \cdot \{[(K_b \cdot \overline{K}_c) \vee \overline{K}_b] \cdot [(K_b \cdot \overline{K}_c) \vee K_c]\}$

12. $\overline{K}_a \cdot \{[(K_b \vee \overline{K}_b) \cdot (\overline{K}_b \vee \overline{K}_c)] \cdot [(K_b \vee K_c) \cdot (\overline{K}_c \vee K_c)]\}$

13. $\overline{K}_a \cdot \{[1 \cdot (\overline{K}_b \vee \overline{K}_c)] \cdot [K_b \vee K_c) \cdot 1]\}$

14. $\overline{K}_a \cdot [(\overline{K}_b \vee \overline{K}_c) \cdot (K_b \vee K_c)]$

15. $\overline{K}_a \cdot (\overline{K}_b \vee \overline{K}_c) \cdot (K_b \vee K_c)$ [simplifed version of F]

Appendix B

Many factors influence the growth of an indexing vocabulary, and while estimates of vocabulary growth are difficult to make they are possible to do if we take into consideration certain observed processes which occur in the development of information retrieval systems.

Each indexing term in the vocabulary is assigned to documents within the data base a certain number of times. If we take these individual term assignment frequencies and rank them from the highest to the lowest values, we often find that they conform to a hyperbolic or Zipfian [23] distribution [Van Rijsbergen, Arthur D. Little]. Thus,

$$N_A = \sum_{i=1}^{N_T} F_i = F_1 \ (\ln \ (2N_T + 1) - 0.116)$$

Where N_A = the total indexing assignments in the data base, N_T = the total number of unique terms in the vocabulary, F_i is the assignment frequency of term i, and F_1 is the assignment frequency of the most frequently assigned subject term in the data base (i.e., term of frequency rank 1).

In our example, since we know the mean depth of indexing is 6 and the number of documents in the data base is 10,000, we can estimate N_A with the following equation [derived from Bird]:

$$N_A = \sum_{i=1}^{t} t \ (e^{-m}) \ (\frac{m^i}{i!}) \ (N_D)*$$

Where t = the maximum number of vocabulary terms assigned to any document in the data base (with a mean depth of 6 we would expect a maximum depth of about 14 or 15), m = the mean depth (here, 6), and N_D = the number of documents in the collection (here, 10,000). Setting t = 15, N_A = 59,800 (this will be the number of tuples in the KEYWORDS relation]. Now, by substitution:

$$N_A = 59,800 = F_1 \ (1_n \ (2N_T + 1) - 0.116)$$

Because the rank-frequency distribution is hyperbolic, then if the distribution were perfect, F_1 would be equal to N_T. In empirical studies it was found that F_1 is somewhat less than a perfect distribution would predict, and N_T somewhat greater [Groos]. If we solve for a value of N_T slightly greater than F_1 we find that:

$N_T = 7,000$

$F_1 = 6,350$

As it turns out, 7,000 will be an estimate of the maximum reasonable value for N_T. The Zipfian distribution is an accurate model for the growth of indexing assignments when new subject terms are added to the system vocabulary at a fairly constant rate. This is the case for the early stages of data base growth, and for all growth in data bases which cover areas like chemical research, pharmacology or patents. For most data bases, though, the term assignment frequency distribution is Zipfian only in the early stages, and the addition of new vocabulary terms falls off as new documents are added to the data base [Lancaster, McClelland, Blagden]. This is because most of the new documents deal with the same subjects that older documents on the data base deal with. This kind of vocabulary growth is log-normal rather than Zipfian and is best modeled by [Wall]:

$$N_T = [3{,}000 \log_{10} (N_A + 7{,}100)] - 11{,}000$$

With N_A = 59,800, then the predicted size of the vocabulary would be 3,480. Thus, we can estimate that the likely size of our indexing vocabulary (N_T) would be between 3,480 and 7,000 terms (where N_D = 10,000 and mean depth = 6).

* For small values of m, N_A can be estimated more easily as the product m x N_D. This approximation becomes less accurate as m increases.

Appendix C

The number of co-occurring index terms in a data base can be estimated in much the same way as index term assignments were estimated (Appendix B). With 10,000 documents and a mean indexing depth of 6, the approximate total number of index term co-occurrences (N_C) can be determined by using the following equation [derived from Bird]:

$$N_C = \sum_{i=1}^{t} P_2^t \, (e^{-m}) \, \left(\frac{m^i}{i!}\right) \, (N_D)$$

Where t = the maximum number of vocabulary terms assigned to any document in the data base (here, 15), m = the mean depth (here, 6), and N_D = the number of documents in the collection (here, 10,000). Setting t = 15, N_C = 358,411.

N_C is the total number of co-occurrences which have occurred in the data base index term assignments, but the number of tuples estimated to exist in the THESAURUS relation is equal to the number of unique co-occurrences of terms which exist in the data base (i.e., no matter how many times (>0) index terms T_i and T_j are both assigned to the same documents it will require only two tuples in the THESAURUS relation to model their relationship).

Unlike index term assignments, there are no comparable studies of how term co-occurrences are distributed. It is not unreasonable, though, to assume that the character of the distribution of index term co-occurrences is similar to the distribution of index term assignments. By assuming that the distribution of term co-occurrences is hyperbolic or Zipfian, we can estimate the number of unique co-occurrences which should occur in our hypothetical data base.

N_C is comparable to N_A (Appendix B) and can be substituted for it in the equation we used to represent the distribution of index term assignments:

$$358{,}411 = F_1 \, (\ln (2N_T + 1) - 0.116)$$

We can reinterpret F_1 as the number of co-occurrences of the most frequently co-occurring term on the data base, and we can reinterpret N_T as the number of uniquely occurring term pairs (or, co-occurrences). Solving the above formula for equal values of F_1 and N_T we find that $N_T = F_1 = 32{,}600$. (Unlike our solution for index term assignments we have no evidence that F_1 will be somewhat less than N_T). N_T, of course is the value which represents the number of tuples estimated to be in the THESAURUS relation.

Unlike index term assignments, we would not expect the number of new unique term co-occurrences to fall off as markedly as the number of new terms added to the vocabulary does during data base growth. This is because

even if new terms are not added to the vocabulary, new pair combinations can be generated almost indefinitely (with a vocabulary of between 3,480 and 7,000 terms (Appendix B), the total number of possible unique pair combinations is:

$$P_2^{3,480} \longrightarrow P_2^{7,000} = 10,854,400 \longrightarrow 49,000,000$$

At most, the 32,600 term co-occurrences estimated to occur in the hypothetical data base represent only .3% of the possible terms combinations. Clearly, the growth in the number of new unique term co-occurrences is not rigidly dependent on the addition of new terms to the vocabulary, though there is undoubtedly some relationship between the addition of new vocabulary terms and the occurrence of new term pairings.

Bibliography

1. Arthur D. Little, Inc. Centralization and Documentation. Cambridge, Mass., 1963.

2. Bird, P.R. "The Distribution of Indexing Depth in Documentation Systems." Journal of Documentation, v30:4, December 1974, pp. 381-392.

3. Blagden, J.F. Management Information Retrieval: A New Indexing Language. London: British Institute of Management, 1969. (2nd edition, 1971).

4. Blair, David C. "The Data-Document Distinction in Information Retrieval." Communications of the ACM, v27:4, pp. 369-374, April 1984.

5. Blair, David C. "Searching Biases in Large, Interactive Document Retrieval Systems." Journal of the American Society for Information Science, v31:4, pp. 271-277, July, 1980.

6. Blair, David C. "SQUARE (Specifying Queries As Relational Expressions) as a Document RetrievalLanguage" unpublished working paper, University of California, Berkley, Spring 1974.

7. Chamberlin, D.D., et. al. "SEQUEL 2: A Unified Approach to Data Definition, manipulation and control," IBM J. Research and Development, v20, 1976, pp. 560-575.

8. Codd, E.F. "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, v.13, 1970.

9. Crawford, Robert G. "The Relational Model in Information Retrieval." Journal of the American Society for Information Science, v32:1, pp. 51-64, 1981.

10. Date, C.J. An Introduction to Database Systems. Addison-Wesley, Reading, Mass., 3rd ed. 1981.

11. Dattola, R.T. "FIRST: Flexible Information Retrieval System for Text." Journal of the American Society for Information Science, v. 30:1, pp. 10-14, 1979.

12. Fairthorne, Robert A. "Empirical Distributions (Bradford-Zipf-Mandelbrot) for Bibliometric Description and Prediction." Journal of Documentation, v25:4, December 1969, pp. 319-343.

13. Groos, O.V. "Bradford's Law and the Keenan-Atherton Data." American Documentation v19:1, 1967, p. 46.

14. Jacquesson, Alain and William Schieber, "Term Association Analysis on a Large File of Bibliographic Data, Using a Highly-Controlled Indexing Vocabulary." Information Storage and Retrieval, v. 9, pp. 85-94, 1973.

15. Lancaster, F. W. Vocabulary Control for Information Retrieval. Information Resources Press, Washington, D. C., 1972.

16. Levien, R. E. and M. E. Maron. A Computer System for Inference Execution and Data Retrieval." The RAND Corp., RM-5085-PR, September 1966.

17. Levien, R.E. and M.E. Maron. "Relational Data File: A Tool for Mechanized Inference Execution and Data Retrieval." The RAND Corp., RM-4793-PR, December 1965.

18. McClelland, R.M.A. and W.W. Mapleson. "Construction and Usage of Classified Schedules and Generic Features in Coordinatted Indexing." ASLIB Proceedings, v. 18, pp. 290-299, 1966.

19. Macleod, I.A. "The Relational Model as a Basis for Document Retrieval System Design." The Computer Journal, v. 24:4, pp. 312-315, 1981.

20. Macleod, I.A. "SEQUEL as a Language for Document Retrieval." Journal of the American Society for Information Science, v. 30:5, pp. 243-249, 1979.

21. Van Rijsbergen, C.J. Information Retrieval. Butterworths, London, 1979, 2nd ed.

22. Wall, E. "Further Implications of the Distribution of Index Term Usage." Proceedings of the American Documentation Institute, v1, pp. 457-466, 1964.

23. Zipf, G.K. Human Behavior and the Principle of Least Effort. Cambridge, Mass.: Addison-Wesley, 1949.