Efficiency of Nested Relational Document Database Systems

Justin Zobel James A. Thom Ron Sacks-Davis

Key Centre for Knowledge Based Systems
Department of Computer Science, RMIT
GPO Box 2476V, Melbourne 3001, Australia

Abstract

Systems designed for efficient retrieval of conventional data can be very inefficient at retrieving documents. Documents have more complex structure than conventional data, and the kinds of queries made to document databases are unlike those made to conventional databases. This paper discusses how document storage and retrieval can be effectively supported in a nested relational database system with signature file indexing, and gives a detailed analysis of the space requirements and retrieval times of different document schemas in such a database system.

Keywords: document database, document management, nested relation, query optimisation.

1 Introduction

Conventional relational database systems are designed to support retrieval of information that has simple, repetitive structure. Documents, however, are usually large and have a hierarchical internal structure, as most documents contain several sections, each of which may contain several subsections or other logical units such as paragraphs or tables. Such structures are difficult to efficiently store and retrieve in conventional relational database systems. This is why many systems that support text retrieval, including MINOS (Christodoulakis et al., 1986), MULTOS (Bertino et al., 1988), AIM (Dadam and Lindemann, 1989), and TITAN+ (Thom et al., 1991), are not based on the relational model.

One effective approach to storing documents is in nested (non-first-normal-form) relational database systems (Desai et al., 1987; Roth et al., 1988; Schek and Pistor, 1982). Nested relational systems permit hierarchically structured objects such as documents to be represented in a natural way. Bit-sliced signature file indexing can be used to provide fast access by document content (Kent et al., 1990; Sacks-Davis et al., 1987).

We propose that documents should be further broken into *fragments*, blocks of text holding logical units such as a paragraph. Because fragments are small, they are cheap to retrieve from disc. Use of fragments can increase the size of databases, however, and makes some kinds of queries more expensive to evaluate. In this paper we consider in detail the relative costs of some fragmented and unfragmented nested relational schemes for storing documents, and give formulas by which space and time requirements can be assessed. We use these formulas to estimate optimal fragment sizes for an example document collection.

In Section 2 we discuss fragmentation and describe three ways in which documents can be stored in nested relations. In Section 3 we analyse the space requirements of each of these structures. In Section 4 we discuss the types of queries we expect to encounter, and in Section 5 we discuss costs of these types of queries for each database schema. Related work on document databases is reviewed in Section 6. For reference, a glossary of notation is included at the end of the paper.

2 Storing documents in nested relations

The architecture of any database system should be tailored to the kinds of data to be stored in the system and tailored to how data is to be accessed. For example, in many database systems data access is based on key values, and consequently indexes are designed to facilitate fast key lookup. Data stored in such database systems must have a simple, repetitive structure and each item of the same kind must have the same format. On retrieval, such database systems typically return the tuples with the specified key.

Documents, however, are more loosely structured than conventional data. Even within a document class, individual documents can vary greatly in length, number of sections, number and frequency of terms, and so on. Retrieval is often based on finding documents containing specified terms (that is, retrieval is on content of the document) rather than on primary keys such as document identifiers or secondary keys such as subject codes.

Hierarchically structured objects such as documents can effectively be stored in nested relational database systems (Desai et al., 1987; Roth et al., 1988; Schek and Pistor, 1982). Signature file indexing schemes can be used to index on terms and term pairs occurring in the body of the document, permitting queries on document content; for relations with large numbers of tuples, bit-sliced indexes can be used to minimise index look-up costs (Kent et al., 1990; Sacks-Davis et al., 1987). Signature file indexes consist of a signature for each tuple in the database to be indexed; the length of each signature is proportional to the largest number of distinct terms in a tuple in the database. In this paper we assume a bit-sliced signature file scheme based on multiple organisations (Kent et al., 1990). For such an indexing scheme, unlike inverted file indexing schemes, answering queries does not become more costly as the number of query terms increases.

Documents might be stored in a nested relational database as follows: each document could be represented as a single tuple in which the set of sections is a nested table and each section contains a nested table of subsections. In such a scheme, however, if the unit of retrieval is a tuple, entire documents must be retrieved in response to queries. Moreover, queries on more than one term can match documents in which those terms are widely separated and are probably unrelated. Thus some queries will lead to large amounts of irrelevant material being retrieved. Most importantly, because of the range of document sizes that can occur in a large document collection, bit-sliced signature file indexes can become unacceptably large, making this approach to document storage impractical.

As an alternative, we propose that documents be broken into *fragments*. A fragment is a block of text from a document of a size suitable for display on a terminal, and should consist of a logical unit of text such as a sentence, paragraph, or table. There are several advantages to using fragments. First, the size variation between fragments can be constrained to be far less than the size variation between documents, thus minimising the size of signature files. Second, if a user looks for tuples containing a set of terms, there is some guarantee that the terms occur close together in the retrieved text. Third, use of fragments reduces the volume of disc traffic: retrieving a fragment is considerably cheaper than retrieving an entire document. Fourth, in many applications it is natural to consider documents as consisting of parts rather than as a whole: for example, in hypertext systems documents are represented as parts that are joined by structure, sequence, citation, and other kinds of links (Conklin, 1987; Fuller et al., 1991).

One disadvantage of fragmenting documents is that it can become difficult to find information about the document from which a given fragment was drawn. It is therefore useful to associate title information (document title, author name, and so on) with each fragment. If title information will usually be retrieved with each fragment, it is probably simplest to store the title information with each fragment. If fragmented documents are to be stored in a minimum of space, however, title information should not be repeated. The simplest way to effect this is to store the text in one table and title information in another. To allow a fragment to be joined to its header, and to allow documents to be reconstructed, unique tuple identifiers must be stored with each fragment and each title. Foreign key occurrences of these identifiers can usefully be thought of as pointers to tuples.

As an example of different possible document schemas, consider a document database that holds autopsy reports. The structure of an autopsy report is as follows. The report consists of a case number, the name of the deceased, and several sections with headings and contents. Each section can have several subsections. Three possible schemas for representing autopsy reports are as follows.

Monolithic schema: each document is represented by a single tuple with a nested table of sections, and each section has a nested table of subsections. The text in each section and subsection is stored in a nested table of fragments. This schema is illustrated in Figure 1. In terms of time and space requirements, this is very similar to having no fragmentation.

Segmented schema: each document is represented by a number of tuples, each containing title information, the current section and subsection name, and a single fragment. This schema is illustrated in Figure 2. As can be seen, title information is repeated but the use of foreign keys is avoided.

Duplex schema: each document is represented by a 'title' tuple containing title information and a nested table of sections and subsections. Each subsection includes a nested table of foreign keys of fragments containing the text of that subsection. This schema is shown in Figure 3. Pointers (foreign keys) have been introduced to facilitate movement between fragments and titles.

The separation of text and other information in the duplex schema means that the structure of documents can be explored without any text having to be retrieved; in the monolithic schema, document structure was embedded. We will refer to databases with a segmented or duplex schema as fragmented. Note that the duplex schema is similar to a monolithic schema in a database system in which documents are represented as complex objects, and parts and subparts of objects can be accessed independently.

The schemas described above assume that each document in a collection will have a given structure. More flexible nested relational schemas can also be designed, permitting storage of document collections containing documents of arbitrary structure. The use of fragments can be incorporated into such schemas, but we do not consider them in this paper.

3 Space analysis

In this section we compare the space requirements of the schemas described in Section 2. We assume that words are distributed in text according to the *clustering model* (Thom and Zobel, 1991). In this model, the probability that a document or fragment of w words contains a given word t with occurrence probability $p(t)$ is given by

$$p_w(t) = 1 - e^{-\alpha . w^{\beta} . p(t)}$$

The parameters α and β are dependent on the document collection being stored; typical values are 1.03 and 0.937 respectively (these values are derived from the King James version of the Bible). Initial investigations extending this model indicate that the probability that a document or fragment of w words contains all of the terms $t_1, \ldots, t_m$ can be approximated by

$$p_w(t_1, \ldots, t_m) = \prod_{i=1}^{m} p_w(t_i)$$

where the terms $t_1, \ldots, t_m$ are assumed to be independent. An approximation to the number of distinct terms in a document or fragment of w terms is given by

$$W_f(w) = \sum_{i=1}^{W_d} p_w(t_i)$$

where W_d is the number of distinct terms in the database and the probability $p(t_i)$ of the ith-ranked term is assumed to follow the Zipf distribution

$$p(t_i) = \frac{1}{(log_e W_d + \gamma).i}$$

where γ = 0.5772 is the Euler-Mascheroni constant (Witten and Bell, 1990). Throughout this paper we assume that W_d is 50,000, a typical vocabulary for a large document collection. For the reader's reference, some typical values of $W_f(w)$ are shown in Table 1.

No. of terms w	10	100	1,000	10,000
$W_f(w)$	8.49	62.6	419	2,540

Table 1: Typical values of $W_f(w)$

Document databases can be described by several parameters.

- The number of documents stored is denoted by N, assumed to be 100,000, the number of words of title information is N_T, assumed to be 50, the number of section and subsection headings is H, assumed to be 10, the number of terms in each heading is N_H, assumed to be 5, the depth of nesting of headings is D, which is 2 in our example, and the average number of words in the text of each document is N_t, assumed to be 10,000. Each term is N_b bits long, assumed to be 50. This implies that 5.88 gigabytes of data is to be stored.
- In segmented and duplex databases, documents are divided into F fragments. Each fragment contains on average $N_{ft} = \frac{N_t}{F}$ terms.
- Multi-organisational bit-sliced signature file methods are used to index data, and all fields (other than foreign keys) are indexed. The number of index terms in a relation is the average number of distinct terms in each tuple times the number of tuples in the relation. The size in bits of the signature file for a relation is R times the number of index terms in the relation, where R is assumed to be 32.
- Foreign keys occupy P bits, assumed to be 32.

In such a collection $p(t)$ would typically range from about 10^{-8}, for words that only occur a few times (note that Zipf's formula substantially overestimates this value), to about 5×10^{-2}, for words such as *the*. Query terms tend to be less common terms, so that the most common query terms would have $p(t) \approx 10^{-4}$.

Approximate space requirements are given by the following formulas.

Tuple size (bits):

Monolithic	$S_m = (N_T + H.N_H + N_t).N_b$
Segmented	$S_s = (N_T + D.N_H + N_{ft}).N_b$
Duplex—titles	$S_t = (N_T + H.N_H).N_b + P.F$
Duplex—fragments	$S_f = N_{ft}.N_b + 2.P$

Data size (bits):

Monolithic	$S_m.N$
Segmented	$S_s.N.F$
Duplex—titles	$S_t.N$
Duplex—fragments	$S_f.F.N$

Index size (bits):

Monolithic	$R.N.W_f(N_T + H.N_H + N_t)$
Segmented	$R.N.F.W_f(N_T + D.N_H + N_{ft})$
Duplex—titles	$R.N.W_f(N_T + H.N_H)$
Duplex—fragments	$R.N.F.W_f(N_{ft})$

Note that these approximations assume that each document is of roughly the same size. A more accurate estimate of the size of indexes would be based on the largest number of distinct terms in a tuple in a relation rather than on the average number of distinct terms, so that the formulas given above will tend to underestimate the size of indexes, and in particular the size of indexes of monolithic databases. For example, if documents ranged up to 100,000 terms in length with an average length of 10,000 terms, the size of the monolithic index would be about five times greater than that given by the above formula.

In Figure 5 we show how the estimated sizes of indexes and databases varies as fragment size varies. Segmented databases are the largest, but, not surprisingly, the difference decreases as the size of fragments grows. One of the main differences in the space requirements is in the size of the indexes. Recall, however, that this estimate optimistically assumes that each document in the collection is of the same size.

Space requirements can be reduced if data is compressed (Moffat and Zobel, 1991; Witten et al., 1991). However, compression does not affect index size: it reduces N_b only, to perhaps 15 in our example. We do not show results for compressed databases in this paper, but compression favours monolithic structuring in space and fragmented structuring in time.

4 Types of queries

There are many ways in which users might search for documents in a document database system. For example, users may request documents pertaining to a subject or set of terms, or may request the document with a given title and set of authors. Moreover, for imprecise queries neither 'all solutions' nor 'first solution' query evaluation strategies are appropriate: some documents will closely match the query whereas others will only be a poor fit. For example, one appropriate query evaluation technique for imprecise queries is to rank retrieved items on the basis of relevance to the query and return the most closely matched items in order of importance (Salton, 1989).

We will analyse three ways in which documents might be accessed:

By content of document: users can search for a document on the basis of terms occurring in the document. Such queries could be exact match ('find all documents containing the terms *female* and *caucasian*'), or they could be imprecise ('find documents about *caucasian females*'). Documents retrieved in response to imprecise queries must be ranked as discussed above.

By content of fragment: rather than the system returning entire documents that match a (exact or imprecise) query, it can return the appropriate parts of the document. In this strategy, monolithic documents which contain all of the query terms, but in which the terms are widely separated, should not be returned. We believe that this kind of query will be the most common: users who request text containing the terms *female* and *caucasian* are likely to only be interested in documents in which these terms are in, say, the same sentence or paragraph.

By title: users can search for title information on the basis of terms occurring in a title or authors' names. (A similar query type that we do not analyse is queries on section or subsection headings.)

We do not analyse other kinds of queries, such as access by browsing, that a document retrieval system would provide.

5 Query costs

In this section we compare costs of queries on monolithic, segmented, and duplex databases. Query costs for fragmented databases are very different to query costs for monolithic databases. In monolithic databases each retrieved tuple is large, and in general more irrelevant tuples are retrieved since the query terms may not occur close together in the retrieved text. On the other hand, in fragmented databases each retrieved tuple is small, and fewer irrelevant tuples are retrieved. However, some queries on fragmented databases will have join costs that would not exist in monolithic databases.

We now consider costs of typical queries to databases with the structures described in Section 2. We make the following assumptions in addition to the assumptions made in Section 3.

- The unit of retrieval is a tuple.
- Monolithic documents are stored contiguously on disc, and that the fragments of a duplex document are stored contiguously on disc. The latter assumption minimises seek times when several fragments are retrieved from one document. We denote seek+latency time by T_s, and assume an average of 10^{-2} seconds.
- There is a cost associated with retrieving and processing each bit of data. We denote this cost by T_d, and assume 10^{-6} seconds per bit. Processing cost include checking that retrieved tuples are valid (signature methods can return a small percentage of false matches) and processing text into a format appropriate for display. Similarly, we assume that there is a cost T_i of retrieving and processing each bit of index, and assume 10^{-7} seconds per bit.
- Each term sets K bits in the signature of the tuple containing that term, where K is assumed to be 8, and signatures are formed for blocks of tuples rather than individual tuples. Block size is B, assumed to be 32. With these assumptions, the cost of looking up a signature file of a relation of T tuples is $K.(T_s + T_i.\frac{T}{B})$, independent of the number of query terms.

Costs of queries can be estimated as described in the rest of this section. For each query type, we assume that documents or fragments with terms $t_1, \ldots, t_m$ are to be retrieved, where $m \geq 1$.

Query by title

Consider queries on title information such as title or author. Each tuple in the title relation of a duplex database holds all of the title information held in each tuple in the equivalent monolithic schema, but tuples in the title relation (which consist of title information and foreign keys of fragments) are much smaller. For segmented databases, we assume that a bitmap (of size $N.F$ bits) is used to indicate whether each tuple is the first tuple of a document, so that other tuples of the document can be ignored. Approximate costs for retrieval on title are given by the following formulas.

Monolithic

Time to look up index

$I = K.(T_s + T_i.\frac{N}{B})$

Number of matching tuples

$M = p_{N_T}(t_1, \ldots, t_m).N$

Total time

$I + T_s.M + T_d.S_m.M$

Segmented

Time to look up index

$I = K.(T_s + T_i.\frac{N.F}{B})$

Number of matching tuples

$M = p_{N_T}(t_1, \ldots, t_m).N$

Total time

$I + T_s.M + T_d.S_s.M$

Duplex

Time to look up index

$I = K.(T_s + T_i.\frac{N}{B})$

Number of matching tuples

$M = p_{N_T}(t_1, \ldots, t_m).N$

Total time

$I + T_s.M + T_d.S_t.M$

These estimates of retrieval times, for single term queries, are illustrated in Figure 4, which shows how retrieval time varies with fragment size for $p(t) = 10^{-6}$. The number of tuples retrieved is the same for each schema, and is therefore not shown. As can be seen, for almost all fragment sizes queries to the duplex database are faster than to the segmented structure, and much faster than to the monolithic structure.

Query by content of fragment

We believe that in document database systems, in many applications the most common kind of query will be to find documents on the basis of content. We first consider the costs of querying on the basis of document fragment. Note that some documents that satisfy a query will contain no fragments that satisfy the query, because the query terms are widely separated in the document. Also, because fragments are stored contiguously, we assume that once a fragment of a document has been retrieved, no seek is required for subsequent fragments from the same document. Thus, in fragmented databases, the number of seeks is at most the number of documents involved in the query, which will be at most the number of documents retrieved from the monolithic database (hence the use of *min* in the following formulas).

Approximate costs for access by content of document fragment are as follows.

Monolithic

Time to look up index

$I = K.(T_s + T_i.\frac{N}{B})$

Number of matching tuples

$M = p_{N_t}(t_1, \ldots, t_m).N$

Total time

$I + T_s.M + T_d.S_m.M$

Segmented

Time to look up index

$I = K.(T_s + T_i.\frac{N.F}{B})$

Number of matching tuples

$M = p_{N_{ft}}(t_1, \ldots, t_m).N.F$

Total time

$I + T_s.min(M, p_{N_t}(t_1, \ldots, t_m).N) + T_d.S_s.M$

Duplex

Time to look up index

$I = K.(T_s + T_i.\frac{N.F}{B})$

Number of matching tuples

$M = p_{N_{ft}}(t_1, \ldots, t_m).N.F$

Total time

$I + T_s.min(M, p_{N_t}(t_1, \ldots, t_m).N) + T_d.S_f.M$

In a variant of this type of query, title information as well as the retrieved text is displayed to the user, providing contextual information about the text. In monolithic and segmented structures, this information is retrieved in any case, but extra operations are needed in duplex schemas, to retrieve the tuples of title information. The additional costs are as follows.

Duplex variant—additional costs

Time to look up index

$I = K.(T_s + T_i.\frac{N}{B})$

Number of matching tuples

$M = \min(p_{N_t}(t_1, \ldots, t_m), p_{N_{ft}}(t_1, \ldots, t_m).F).N$

Total additional time

$I + T_s.M + T_d.S_t.M$

Times and numbers of matching tuples for access by content of fragment, for single term queries on terms with occurrence probability $p(t) = 10^{-7}$, are shown in Figure 6, and, for single term queries on terms with $p(t) = 10^{-6}$, are shown in Figure 7. Times and numbers of matching tuples for access by content of fragment, for multi-term queries on three terms each with $p(t) = 10^{-4}$, are shown in Figure 8. Again, it was shown that (non-variant) queries on the duplex database are cheaper than queries to the other structures, and are much cheaper than queries to the monolithic structure. As the number of query terms increases, the relative cost of using the monolithic structure increases drastically, as illustrated by the 10,000-fold difference in times between the fragmented and monolithic schemas in Figure 8. For queries with a larger number of terms, the difference is even greater.

In the top graph in Figure 6, it can be seen that, in duplex databases, fragments of about 85 terms have the minimum retrieval time for queries on terms with probability 10^{-7}. This minimum depends on the probability of the query term: as can be seen in Figure 7, for query terms with probability of 10^{-6}, the optimal fragment size is about 25 terms. Note that reducing fragment size makes queries to segmented structures cheaper to evaluate, but, as can be seen in Section 3, at a considerable space penalty.

If variant queries are expected to be common, then the segmented schema is preferable. However, in many applications requests for title information might only be made for a small proportion of the returned fragments, in which case either segmented or duplex schemas would be suitable. For example, if returned fragments are to be ranked, only the fragments to be displayed (usually a small proportion of the total) will require title information.

Note that, as discussed in Section 4, some retrieved monolithic documents may have to be discarded because the query terms are not near to each other in the text of those documents. In some cases, as illustrated in Figure 8, the number of documents to be discarded can become very large. Use of a secondary index that allowed access to monolithic documents on the basis of fragments of the documents would eliminate this problem, at the cost of extra space to store the index. Even with this optimisation, queries to the monolithic schema would still be slower than queries to the other schemas, because of the larger amount of data to be retrieved.

Query by content of document

In monolithic databases, querying by content of document is identical to querying by content of document fragment. In fragmented databases, complex evaluation strategies are required. One strategy is to retrieve all fragments that contain any of the query terms, use them to determine which documents contain all of the query terms, and then retrieve all of the information for that document. To retrieve a whole document, the locations of the first and last fragments must be found, as well as the title of the document in the duplex case. All of the data stored between the first and last fragments of a document should be retrieved, as each document is stored contiguously on disc. Approximate costs are as follows.

Monolithic

As for access by content of document fragment.

Segmented

Time to look up index

$$I = K.m.(T_s + T_i.\frac{N.F}{B}) + 2.K.(T_s + T_i.\frac{N.F}{B})$$

Number of matching tuples

$$M = (p_{N_{ft}}(t_1) + \ldots + p_{N_{ft}}(t_m)).N.F + p_{N_t}(t_1, \ldots, t_m).N.F$$

Total time

$$I + T_s.((p_{N_{ft}}(t_1) + \ldots + p_{N_{ft}}(t_m)).F + p_{N_t}(t_1, \ldots, t_m)).N + T_d.S_s.M$$

Duplex

Time to look up index

$$I = K.m.(T_s + T_i.\frac{N.F}{B}) + K.(T_s + T_i.\frac{N}{B}) + 2.K.(T_s + T_i.\frac{N.F}{B})$$

Number of title tuples

$$M_t = p_{N_t}(t_1, \ldots, t_m).N$$

Number of fragment tuples

$$M_f = (p_{N_{ft}}(t_1) + \ldots + p_{N_{ft}}(t_m)).N.F + p_{N_t}(t_1, \ldots, t_m).N.F$$

Total time

$$I + T_s.((p_{N_{ft}}(t_1) + \ldots + p_{N_{ft}}(t_m)).N.F + 2.M_t) + T_d.(S_t.M_t + S_f.M_f)$$

Costs given by these formulas are shown in Figure 9, for queries on three terms each with $p(t) = 10^{-4}$. Compared to monolithic databases, this type of query is marginally more expensive in duplex databases, and substantially more expensive in segmented databases.

In an environment in which this kind of query is expected to be common, a monolithic structure would be superior. However, we believe that this kind of query would be rare. For example, in hypertext systems users deal almost exclusively with parts of documents, and would rarely or never query a whole document.

6 Related work

These approaches to document management have been developed as a consequence of experience with document databases and TITAN+, a research prototype nested relational database system developed at the Key Centre for Knowledge Based Systems in Melbourne, Australia (Thom et al., 1991). TITAN+ uses multi-organisational bit-sliced signature file indexing to provide access by content (Kent et al., 1990; Sacks-Davis et al., 1987). Further discussion of the TITAN+ features appropriate to document management can be found elsewhere (Sacks-Davis et al., 1990). Faloutsos compares signature methods with other access methods, including inverted files, and also considers effects such as clustering (Faloutsos, 1985). Other nested relational database systems that support text retrieval include AIM (Dadam and Lindemann, 1989) and DASDBS (Schek et al., 1990).

Methods for arriving at optimal nested relational designs are considered by Hafez and Ozsoyoglu (Hafez and Ozsoyoglu, 1988). For example, once a fragment size had been chosen, their methods could be used to derive the duplex structure given the monolithic structure and a series of sample queries. However, these methods cannot be used to derive information about fragment sizes, nor can they be used to identify cases in which information should be repeated, such as in segmented schemas.

Another database system designed for document management is MULTOS (Bertino et al., 1988). In contrast to our fragmented schemes, documents are stored as single entities and the underlying storage organisation is not based on the nested relational model. In MINOS, which is also designed for document management, documents are represented as complex objects with explicit structure (Christodoulakis et al., 1986). However, like MULTOS, MINOS stores documents in a monolithic structure. Mistral/11 is an early document retrieval system, and was based on the relational model (Macleod, 1981). Mistral/11 also uses monolithic structures.

7 Conclusion

The major conclusion of this paper is that documents should be broken into fragments, each of which should be stored in a separately. There are several reasons for this. First, in many applications most queries will be on title or fragment content; fragmentation permits much faster access to the data for such queries, because in general much less data is retrieved from a fragmented database than from an monolithic database. Our results indicate that for large document collections queries to monolithic schemas are so slow that such schemas are impractical. Second, documents can vary greatly in length, which can cause bit-sliced signature file indexes to become unreasonably large. In fragmented databases, size variations are contained so that this problem does not arise. Third, fragments are similar to the units of text handled by some document database applications, for example hypertext.

We have described two possible fragmented schemas, a segmented schema and a duplex schema. For both of these schemes, we have analysed the relationship between fragment size, database size, and query response time. There is no fixed optimal fragment size, but in general smaller fragments give better retrieval time. However, a good fragment size would be such that each fragment contains a sentence or paragraph. Of the two schemas for fragmented databases, the duplex schema occupies substantially less space, and is faster for most of the query classes we have considered. Choice of schema will depend on the application, but we expect that the duplex schema would generally be preferred.

There are many variants of these schemes that might be considered: use of secondary indexes to provide access to fragmented databases on the basis of the content of documents rather than the content of fragments; merging the segmented and duplex schemas to get better retrieval speed; and considering the costs of further query types. However, such investigations would not extend our fundamental result: that fragmentation permits much faster access to data stored in large document databases.

Acknowledgments

We would like to thank Kotagari Ramamohanarao and Alan Kent for their assistance with this paper. This work was partly supported by an Australian Research Council grant.

References

Bertino, E., Rabitti, F., and Gibbs, S. (1988). Query processing in a multimedia document system. *ACM Transactions on Office Information Systems*, 6(1):1–41.

Christodoulakis, S., Theodoridou, M., Ho, F., Papa, M., and 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.

Conklin, J. (1987). Hypertext: An introduction and survey. *IEEE Computer*, 20(9):17–41.

Dadam, P. and Lindemann, V. (1989). Advanced information management (AIM): Advanced database technology for integrated applications. *IBM Systems Journal*, 28(4):661–681.

Desai, B., Goyal, P., and Sadri, F. (1987). Non first normal universal relations: an application to information retrieval systems. *Information Systems*, 12(1):49–55.

Faloutsos, C. (1985). Access methods for text. *ACM Computing Surveys*, 17(1):49–74.

Fuller, M., Kent, A., Sacks-Davis, R., Thom, J., Wilkinson, R., and Zobel, J. (1991). Querying in a large hyperbase. In *Second International Conference on Database and Expert Systems Applications*, Berlin, Germany. (To appear).

Hafez, A. and Ozsoyoglu, G. (1988). The partial normalised storage model of nested relations. In Bancilhon, F. and DeWitt, D., editors, *Proceedings of the Fourteenth International Conference on Very Large Databases*, pages 100–111.

Kent, A., Sacks-Davis, R., and Ramamohanarao, K. (1990). A signature file scheme based on multiple organisations for indexing very large text databases. *Journal of the American Society for Information Science*, 41(7):508–534.

Macleod, I. (1981). A database management system for document retrieval applications. *Information Systems*, 2(1):131–137.

Moffat, A. and Zobel, J. (1991). High performance compression of large document databases. Technical Report 37, Key Centre for Knowledge Based Systems, Departments of Computer Science, RMIT and the University of Melbourne, Melbourne, Australia.

Roth, M., Korth, H., and Silberschatz, A. (1988). Extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems*, 13(4):289–417.

Sacks-Davis, R., Kent, A., and Ramamohanarao, K. (1987). Multi-key access methods based on superimposed coding techniques. *ACM Transactions on Database Systems*, 12(4):655–696.

Sacks-Davis, R., Wen, W., Kent, A., and Ramamohanarao, K. (1990). Complex object support for a document database system. In *Proceedings of the Thirteenth Australian Computer Science Conference*, pages 322–333.

Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley, USA.

Schek, H.-J., Paul, H.-B., Scholl, M., and Weikum, G. (1990). The DASDBS project: Objectives, experiences, and future prospects. *IEEE Transactions on Knowledge and Data Engineering*, 2:25–43.

Schek, H.-J. and Pistor, P. (1982). Data structures for an integrated data base management and information retrieval system. In *Proceedings of the Eighth International Conference on Very Large Data Bases*, pages 197–207.

Thom, J., Kent, A., and Sacks-Davis, R. (1991). TQL: A nested-relational query language. *Australian Computer Journal*. (To appear).

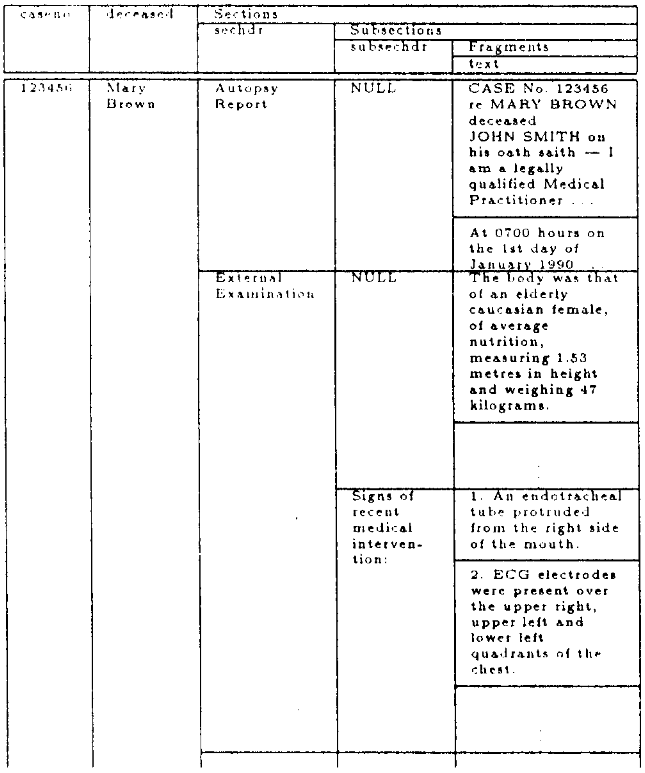
Thom, J. and Zobel, J. (1991). A model for word clustering. Technical Report 39, Key Centre for Knowledge Based Systems, Departments of Computer Science, RMIT and the University of Melbourne, Melbourne, Australia.

Witten, I. and Bell, T. (1990). Source models for natural language text. *International Journal of Man-Machine Studies*, 32:545–579.

Witten, I., Bell, T., and Nevill, C. (1991). Models for compression in full-text retrieval systems. In *Proceedings of the IEEE Data Compression Conference*, pages 23–32.

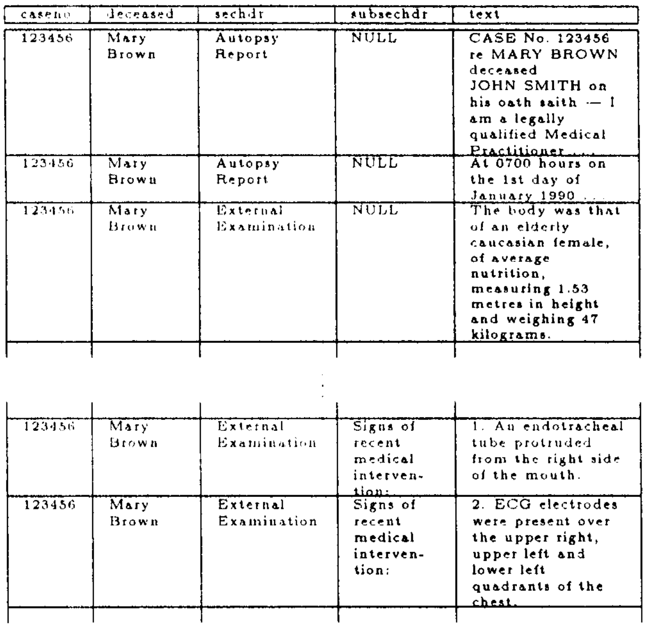
Glossary of notation

Notation	Definition	Value
B	block size	32
D	nesting depth of headings	2
F	no. of fragments per doc.	
H	no. of section headings	10
K	no. of bits set by term in signature	8
I	seconds to search index	
m	no. of query terms	
M	no. of matching tuples	
N	no. of docs	100,000
N_b	no. of bits per term	50
N_{ft}	no. of terms per frag.	
N_H	no. of terms in heading	5
N_t	no. of terms per doc.	10,000
N_T	no. of terms of title information	50
$p(t)$	prob. that term t occurs	
$p_w(t_1, ..., t_m)$	prob. that frag. or doc. of length w contains all of the terms $t_1, \ldots, t_m$	
P	size of foreign keys	32
R	no. of bits in signature per term	32
S_f	fragment tuple size in bits	
S_m	monolithic tuple size in bits	505,000
S_s	segmented tuple size in bits	
S_t	title tuple size in bits	
T_d	seconds to process one bit of data	10^{-6}
T_i	seconds to process one bit of index	10^{-7}
T_s	seconds per seek+latency	10^{-2}
W_d	no. of distinct term in db	50,000
$W_f(w)$	no. of distinct terms in doc. or frag. of w terms	



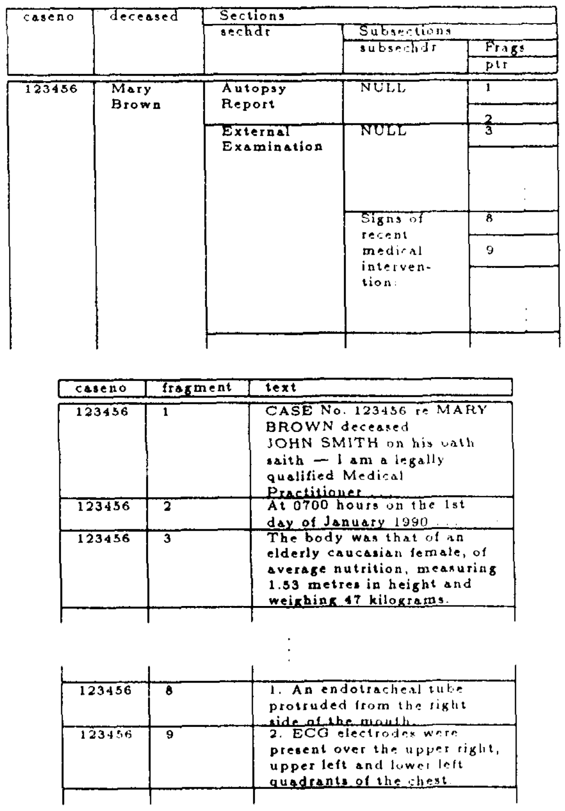
caseno	deceased	Sections		
		sechdr	Subsections	
			subsechdr	Fragments
				text
123456	Mary Brown	Autopsy Report	NULL	CASE No. 123456 re MARY BROWN deceased JOHN SMITH on his oath saith — I am a legally qualified Medical Practitioner . . .
				At 0700 hours on the 1st day of January 1990 . .
		External Examination	NULL	The body was that of an elderly caucasian female, of average nutrition, measuring 1.53 metres in height and weighing 47 kilograms.
				⋮
			Signs of recent medical interven-tion:	1. An endotracheal tube protruded from the right side of the mouth.
				2. ECG electrodes were present over the upper right, upper left and lower left quadrants of the chest.

Figure 1: A schema for monolithic storage of documents



caseno	deceased	sechdr	subsechdr	text
123456	Mary Brown	Autopsy Report	NULL	CASE No. 123456 re MARY BROWN deceased JOHN SMITH on his oath saith — I am a legally qualified Medical Practitioner . . .
123456	Mary Brown	Autopsy Report	NULL	At 0700 hours on the 1st day of January 1990 . .
123456	Mary Brown	External Examination	NULL	The body was that of an elderly caucasian female, of average nutrition, measuring 1.53 metres in height and weighing 47 kilograms.
		⋮		
123456	Mary Brown	External Examination	Signs of recent medical interven-tion:	1. An endotracheal tube protruded from the right side of the mouth.
123456	Mary Brown	External Examination	Signs of recent medical interven-tion:	2. ECG electrodes were present over the upper right, upper left and lower left quadrants of the chest.

Figure 2: A schema for segmented storage of documents



caseno	deceased	Sections			
		sechdr	Subsections		
			subsechdr	Frags	
				ptr	
123456	Mary Brown	Autopsy Report	NULL	1	
				2	
		External Examination	NULL	3	
				⋮	
			Signs of recent medical interven- tion:	8	
				9	
				⋮	

caseno	fragment	text
123456	1	CASE No. 123456 re MARY BROWN deceased JOHN SMITH on his oath saith — I am a legally qualified Medical Practitioner . . .
123456	2	At 0700 hours on the 1st day of January 1990 . . .
123456	3	The body was that of an elderly caucasian female, of average nutrition, measuring 1.53 metres in height and weighing 47 kilograms.
⋮	⋮	⋮
123456	8	1. An endotracheal tube protruded from the right side of the mouth.
123456	9	2. ECG electrodes were present over the upper right, upper left and lower left quadrants of the chest.

Figure 3: A schema for duplex storage of documents

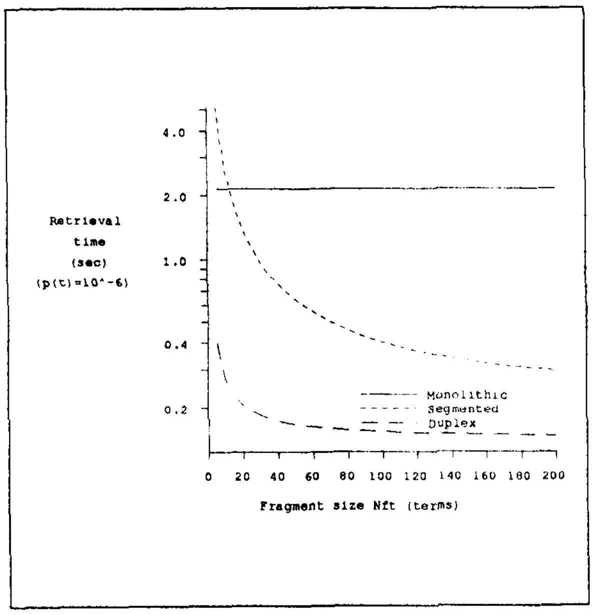
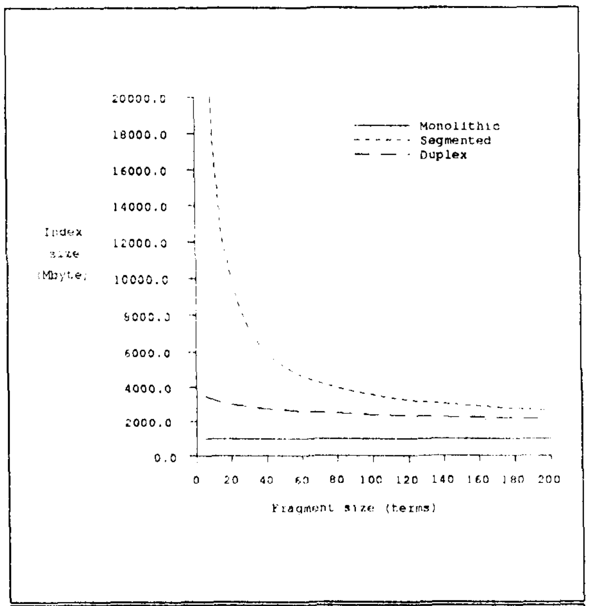


Figure 4: Queries on title by fragment size and by term probability



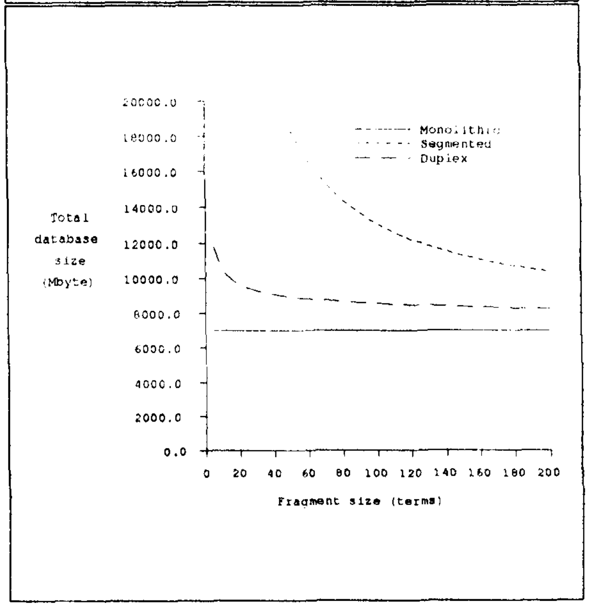
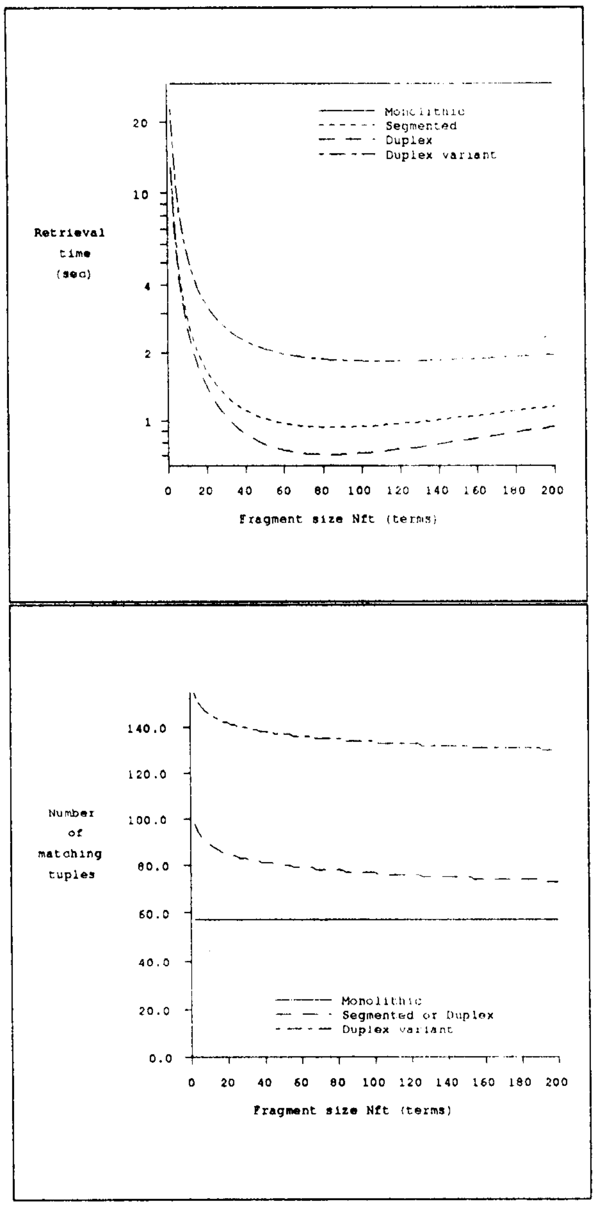


Figure 5: Sizes of indexes and databases by fragment size



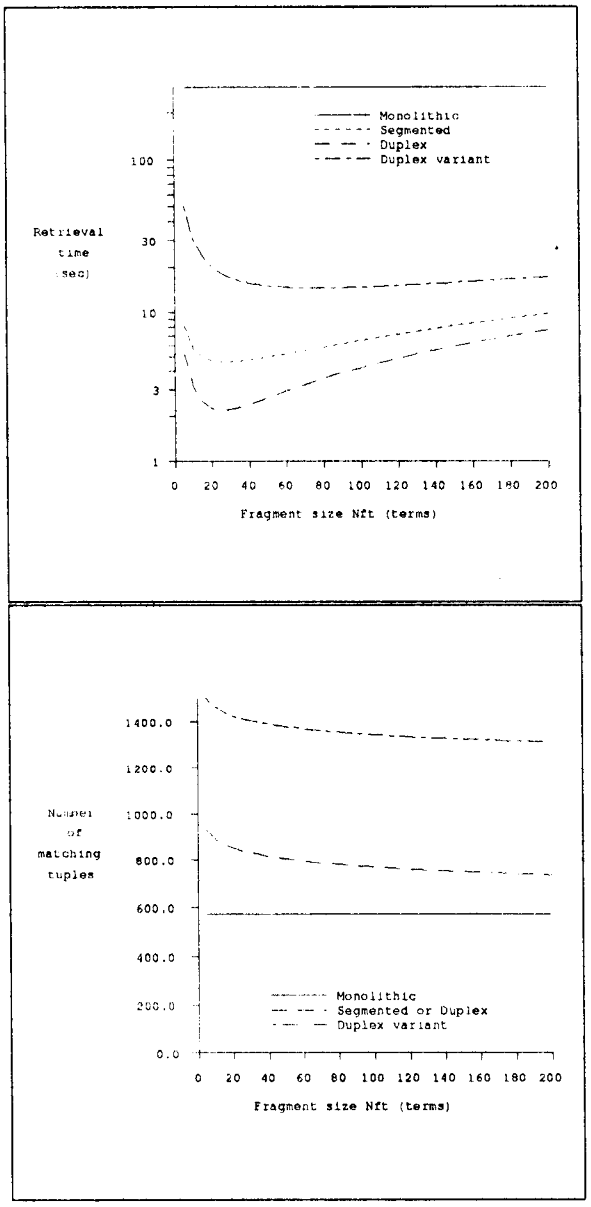


Figure 6: Single-term queries on fragment content by fragment size, $p(t) = 10^{-7}$

Figure 7: Single-term queries on fragment content by fragment size, $p(t) = 10^{-6}$

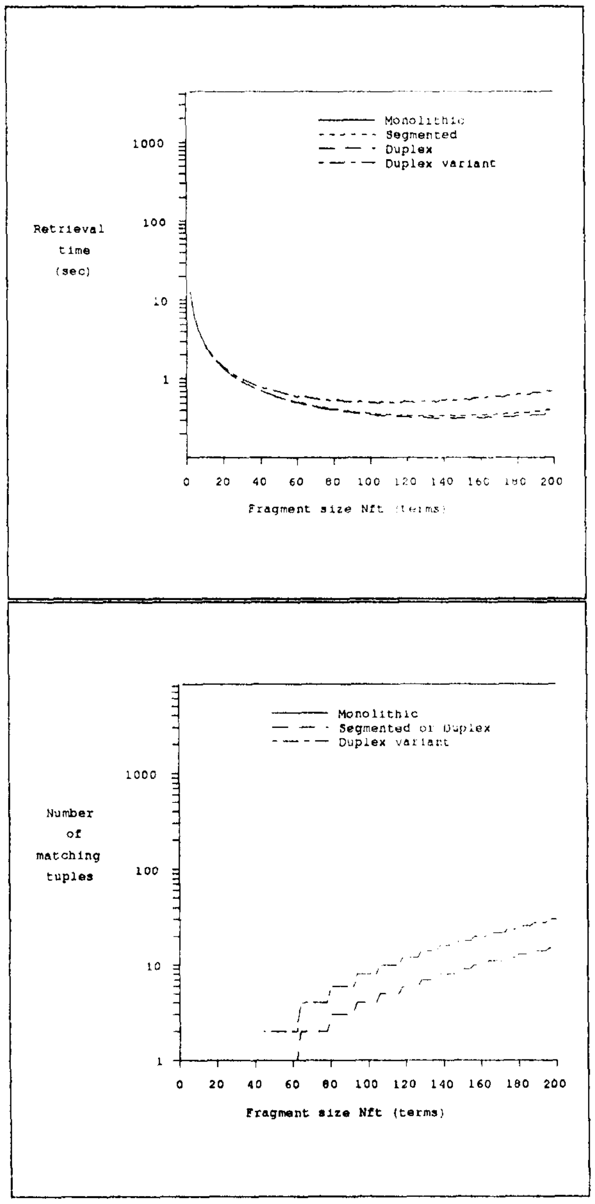


Figure 8: Multi-term queries on fragment content by fragment size, each $p(t) = 10^{-4}$

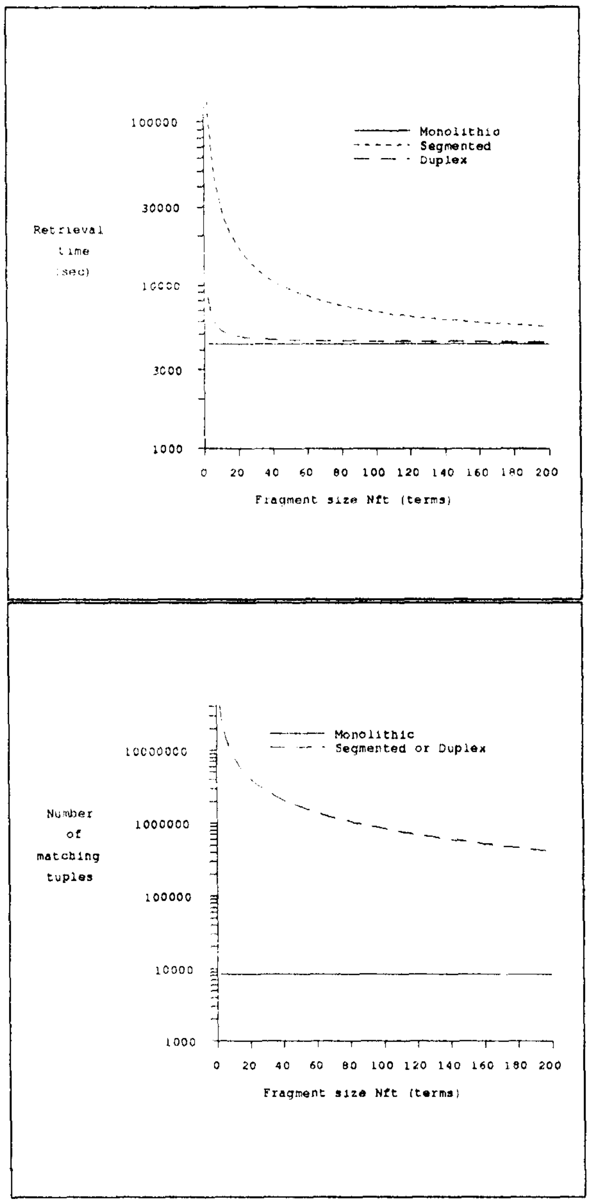


Figure 9: Multi-term queries on document content by fragment size, each $p(t) = 10^{-4}$