In M. Khosrowpour (ed.), *Proceedings of the 5th International Conference of the Information Resources Management Association* (San Antonio, TX, May 22-25, 1994), pp. 448-455. Idea Group Publishing, Harrisburg.

Preprint version

A Comparison of Three Data Models for Text Storage and Retrieval Systems: The Relational Model Revisited

Morten Hertzum
Dept. of Computer Science, University of Copenhagen

A number of software toolkits exist for the development of text storage and retrieval systems (TSARS). This study compares three data models applicable to such toolkits and discusses the suitability of one of them as the basis of a toolkit unifying all three data models. The three data models are: (1) the text model, also known as the inverted file approach, (2) the hypertext model, and (3) the relational model. In the design of the relational model changeability was a key consideration, but more often it is sacrificed to save development resources or improve performance. As it is not uncommon to see successful TSARS exist for 15-20 years and be subject to manifold changes during their lifetime, it is the relational model which is considered for use in the unified toolkit. It seems as if the relational model can be enhanced to incorporate the text model and the hypertext model.

1. Introduction

Text storage and retrieval systems (TSARS) are systems intended to support the organization of and access to bodies of texts. To avoid developing TSARS from scratch, they are mostly based on a more or less application-specific toolkit, i.e. a set of integrated software tools covering large or all parts of an application. The purpose of this study is twofold: (1) To review the strengths and weaknesses of three data models applicable to such toolkits, namely the text model which is also known as the inverted file approach, the hypertext model, and the relational model. (2) To discuss the suitability of the relational model as the basis of a toolkit unifying all three data models. In the literature, the relational model is generally considered inferior to other data models for the development of TSARS (van Rijsbergen, 1979; Lynch & Stonebraker, 1988). From the present author's point of view this reflects an underestimation of the need for adapting TSARS to changes during their lifetime.

In the next section the three data models are introduced and their main properties as well as major aspects of their implementations are discussed, based on the literature. Section 3 briefly describes three commercially available toolkits implementing the data models and, to some extent, combining facilities from them. Section 4 concerns the possibility of developing a toolkit having the major strengths of all three data models. The choice of the relational model as the basis of this toolkit implies that it is expected to be suited for this purpose; it does not imply that the two other data models are judged unsuited.

2. The three data models

TSARS share a number of properties, but there are also fundamental differences which have motivated the development and use of different data models. A couple of the differences are highlighted in the following simple classification which distinguishes two types of systems and three types of text involvement, see figure 2.1 which also includes examples. The two types of systems

are: (1) *Text retrieval systems*, which provide access to fixed or externally updated corpora. (2) *Text filing and retrieval systems*, which provide facilities for both filing and, subsequently, retrieving texts. The three types of text involvement are: (1) *Registering*, in which the system handles fields with information about the texts, sometimes known as text surrogates, but not the text itself. (2) *Storing*, in which the system handles the text itself along with certain attribute fields. However, each text is treated as an atomic entity. (3) *Modelling*, in which the system handles information about the internal structure of the texts along with the text itself and certain attribute fields. Here, the retrieval facilities allow exploitation of the text structure, for example by providing access both to entire texts at a time and to individual parts while retaining their relation to the entire text.

type of system	type of text involvement		
	registering	storing	modelling
text retrieval systems	keyword-based information retrieval systems	full-text information retrieval systems	hypertext 'books'
text filing and retrieval systems	journalization systems	archival systems	electronic publishing

Figure 2.1. A simple classification of TSARS, including examples.

2.1 The text model

The majority of text retrieval systems with little or no modelling of the texts are based on what Macleod (1991) terms the text model. The text model is little more than a generalized description of the way in which these systems are currently implemented and have been implemented for years without major modifications. Texts are conceived of as independent documents with two major components, a set of attributes and a content. The attributes vary with the application area, but bibliographic information, such as title, author, and date of publication, is almost always present. The contents is the text from which the document is composed and may or may not be included in actual systems. To improve performance an index is established containing all the words appearing in the attributes and, if part of the system, in the contents. This index is commonly implemented as an inverted file, hence the designation the inverted file approach (Salton & McGill, 1983; Macleod, 1991).

The major strength of the text model is that it is developed specifically for text storage and retrieval (van Rijsbergen, 1979). The text model allows a reasonable natural representation of the documents, including the possibility to subdivide documents into a sequence of paragraphs. Retrieval can be conditioned on the contents of particular paragraphs and document display can be restricted to specific paragraphs. The Boolean logic query language is syntactically simple, and in combination with the index it provides fast performance even on very large text databases. However, it is well-known that many users fail to understand the semantics of Boolean retrieval and, thus, experience manifold difficulties in dealing with the query language, see e.g. (Borgman, 1986). Furthermore, the text model is easy to implement and most systems include text related facilities, such as data entry tools, automatic creation of the index, and query constructs for phrase handling and proximity searching.

The major weaknesses of the text model are the crude, slow update facilities and the lack of flexibility and extensibility. As argued by Hertzum et al. (1993), one reason for this is that the traditional applications of the text model are text retrieval systems, not text filing and retrieval

systems. This issues in the assumption that the demands on the systems are rather stable and that updates are sufficiently rare to be collected over a period of time and then executed collectively. Some systems have these characteristics, others do not, and still others evolve from having to not having them. With the current emphasis on interactivity and integrated information handling environments the size of the last two categories is likely to be considerable and increasing. The lack of flexibility is also evidenced by most implementations allowing only one document format per application.

A further weakness of the text model is the limited possibilities of modelling inter- and intra-document structure and the exclusive provision of querying as the way to access the documents. This separates the text model fundamentally from the hypertext model.

2.2 The hypertext model

The hypertext model was first described by Bush (1945), Engelbart (1963), and Nelson (1967) in their attempts to devise the ultimate way of interaction between humans and TSARS. With its emphasis on non-linear reading and text organization the hypertext model has evolved around an idea of associative structuring. The characteristic feature of hypertext systems is that the text database is a network of interlinked text chunks. However, contemporary hypertext systems vary considerably in their definitions of the network concept. Recently, the Dexter hypertext reference model (Halasz & Schwartz, 1990) has been proposed in an effort to standardize the terminology and provide a basis for comparing hypertext systems.

The major strengths of the hypertext model are the straightforward representation of texts, the flexible, powerful linking facilities, and the interactive annotation and extension facilities. The emphasis on human-computer interaction has equipped most systems with graphical user interfaces manipulated by mouse-clicking on buttons embedded in the texts and elsewhere on the screen (Conklin, 1987). Typically, hypertext systems support text as well as graphics; thus, figures and the like found in many printed texts are not lost when the text is turned into hypertext. Furthermore, some systems support hypermedia. The hypertext model emphasizes interactivity. This is apparent in the facilities allowing new links and text chunks to be created. However, it is even more apparent in the facilities providing access to the texts. The way to access the texts is browsing, i.e. exploring text chunks and following links between them. To help the users stay oriented while browsing, the links are often supplemented by a visual representation of the network, such as a map or tree.

The major weaknesses of the hypertext model are inadequate query facilities, crude update facilities, and the closedness of most hypertext systems. Browsing is an exploratory search strategy well-suited in situations where understanding is given priority to retrieval (Marchionini & Shneiderman, 1988). The hypertext model practically lacks query facilities to support the situations in which retrieval is the major objective, for instance because the subject is understood but a certain fact forgotten. Usually, the update facilities affect a single link or text chunk at a time and are unsuited for making significant restructurings or global changes. Part of the explanation for the humble query and update facilities is that higher level facilities require the imposition of some structure on the network—and structure is contrary to the ideal of freedom permeating hypertext work (Parsaye et al., 1989). Most hypertext systems are closed systems where texts must be copied into the system before links can be attached to them (Puttress & Guimaraes, 1990). A few systems provide a link service instead and thereby make it possible to create links, for instance, to texts which are currently under preparation in a text processing system.

2.3 The relational model

The relational model was introduced by Codd in 1970 (Codd, 1970). During the 1980s systems implementing it have become dominant for all database applications, except those involving large

amounts of text. However, as TSARS in which the texts are merely registered are rather similar to traditional applications of the relational model, such systems are sometimes relational. In the relational model data is arranged as rows (tuples) in 2-dimensional tables (relations), each tuple consisting of a number of attributes. The relational model has succeeded in providing systems developers with a more declarative, set-at-a-time programming language which leaves the translation into a record-at-a-time access path to the database management systems (DBMS). The goal was to take care of the file handling details of a broad range of applications, not to support all aspects of a certain class of applications (Codd, 1982).

The major strengths of the relational model are its simplicity, flexibility, extensibility, and the powerful data manipulation facilities. The data manipulation facilities, commonly SQL, include a query language as well as update facilities, both of which operating on sets. Links are modelled easily and may reference texts as well as any other object. In the design of the relational model change was considered an essential and unavoidable property of information systems (Codd, 1982). Thus, new relations and new attributes can be added in a piecemeal fashion without affecting existing applications. It is also possible to experiment with the effects of having or not having particular indexes on the relations without affecting the applications in any other way than in performance. Furthermore, relational DBMS include numerous facilities for managing databases, such as recovery routines, performance measuring facilities, and authorization mechanisms; Blair (1988) lists many more such facilities.

The major weaknesses of the relational model are difficulties modelling text with relations, problems achieving satisfactory performance, and the efforts required to build the user interface on top of the relational DBMS. Some TSARS reference the texts at several levels, for example at document level, chapter level, and paragraph level. This makes retrieval from a relational database somewhat cumbersome as relations must be normalized, i.e. all attributes must be atomic. Thus, the texts must be divided into paragraphs to allow retrieval of individual paragraphs, and then a document or chapter can only be retrieved as a sequence of paragraphs, not as one unbroken entity. Preferably, it should be possible to access the texts at different levels of detail depending on the situation. Normalized relations may be normalized further to reduce redundancy, a process dividing the database into a larger number of relations. Having divided the database into a large number of relations, it is typically necessary to combine—join—data from several relations to answer a query. Thus, a lot of *joins* are required, and each *join* is a time-consuming operation (Lynch & Stonebraker, 1988). Date (1986b) admits that there is some truth in regarding normalization as optimizing for update at the expense of retrieval. Furthermore, a decade ago relational DBMS lacked data structures for storing long texts (Codd, 1982); now this problem has been significantly reduced, e.g. Sybase can handle up to 2 Gb of text as a single attribute.

Many critics of basing TSARS on the relational model have focused on the unsuitability of SQL as a query language for the end-user, see for example Morissey et al. (1986) and Macleod (1991). From the present author's point of view, SQL was never intended for this purpose, but as the internal interface between the database and the application program. Because of its generality SQL is much more complicated than the query language of virtually all TSARS need to be. Furthermore, equating the query language with SQL restricts the system to a command line dialogue. However, compared to the text model and the hypertext model it is no doubt a weakness that the relational model provides part of the system only. The user interface to be build on top of the relational DBMS must also implement any text related facilities as the relational model lacks such facilities.

	Text model	Hypertext model	Relational model
Representation of documents	+ reasonably natural representation of documents - usually, only one document format per application	+ natural representation of documents, including graphics or even hypermedia	- difficulties modelling text with normalized relations - problems handling long texts (a strongly decreasing problem)
Data entry	+ automatic indexing and other data entry tools	+ tools to turn text with special markup into hypertext	- no data entry tools for text
Links	- lack of linking facilities	+ flexible, powerful linking facilities	+ versatile linking facilities; links may involve texts and other objects
Query language	+ syntactically simple query language + specific text related functions, such as phrase handling and proximity searching - inflexible query language	- inadequate query facilities	+ flexible, powerful query language - lack of text related query facilities - SQL unsuited as an end-user query language
User interface	- rather predefined, command or menu based user interfaces	+ direct manipulation, graphical user interfaces	- the user interface is not part of the relational model
Performance	+ fast performance, even on very large text databases	+ satisfactory performance	- performance is a bottleneck; large space requirements
Update	- crude, slow update facilities	- crude update facilities	+ high-level update functions, possibly including integrity constraints
Extensibility	- weak on extensibility	+ straightforward annotation facilities - weak facilities for larger extensions	+ extensibility without affecting applications, including openness towards other systems
Application range	- limitation to a single application area, thus little like recovery routines and such	- lack of many database management facilities, such as performance measuring facilities	+ designed to address a broad range of applications

Figure 2.2. Major strengths (+) and weaknesses (-) of the three data models.

3. Examples

Data models are abstractions and often more rigid than the systems implementing them. By briefly describing three commercially available toolkits, the following examples supplement the above discussion and illustrate certain combinations of facilities from different data models. The examples are BRS/Search (version 6.0) based on the text model, Folio Views (version 2.1) based on the hypertext model, and Oracle, including version 1.1 of the special purpose text retrieval module, based on the relational model. The examples summarize a case study in which the toolkits were used to develop three prototypes of a full-text legal information retrieval system comprising 4 Mb of text from Karnov's Lawbook, a leading body of laws in Denmark. The case study was performed by this author and nine graduate students, see Andersen et al. (1992). Figure 3.1 lists the differences between the major strengths and weaknesses discussed in the literature and whose of the toolkits investigated.

BRS/Search is a widespread and comprehensive toolkit for development of TSARS. It has the strengths of the text model and avoids a number of the weaknesses. The Boolean query facilities are available with several user interfaces, menu based as well as command based and basic as well as comprehensive. Querying is supported by a thesaurus facility and supplemented by the possibility of establishing links in or between documents. Furthermore, applications may include documents in different formats. The weaknesses center around the extensibility and the update facilities. Finally, it is apparent from the development of the prototype that the data entry tools are inadequate for nontrivial applications.

Folio Views builds upon the hypertext model in that the texts are divided into chunks, called folios, and retrieval consists of creating and selecting groups of folios, called views. Views can be created by following links and by posing queries. The query facility is based on an inverted file which includes all words in the texts as well as the attributes assigned to the text chunks. The query facility is central to Folio Views, actually links are implemented as static, embedded queries. The toolkit has the strengths of the hypertext model and, in addition, a reasonable query facility. However, the prototype shows that utilizing the possibilities of the toolkit requires careful and heavy use of attributes, and this leads to a inflexible system as both data entry and update facilities are rather crude.

Oracle is a relational DBMS around which a number of special purpose tools have been built. The toolkit used in the case study consisted of the relational DBMS, the text retrieval module SQL*Textretrieval, and the user interface module SQL*Forms. The text retrieval module extends the relational DBMS with facilities comparable to a simple text model system. These facilities include an unsophisticated mapping of text onto relations, the addition of a text related query sublanguage to SQL, a thesaurus facility, and a number of library functions to support applications development. Furthermore, the toolkit supports form based user interfaces well, and the prototype has satisfactory response times. The weaknesses include that: (1) text is limited to 64 Kb chunks, and handling texts consisting of multiple chunks is the responsibility of the applications developers; (2) the versatile facilities for modelling links are not supported by facilities for link following in the user interface; and (3) the data entry tools are inadequate.

	BRS	Folio Views	Oracle
Representation of documents	+ an application may include several document formats	+ strong facilities for handling attributes assigned to the texts	- text limited to 64 Kb chunks
Data entry	- inadequate data entry tools	- inadequate data entry tools	- inadequate data entry tools
Links	+ reasonable linking facility		
Query language	+ a thesaurus facility	+ reasonable query facilities	+ SQL is extended with a text related query sublanguage + several text related facilities, including word indexes and thesaurus
User interface	+ user interfaces for different kinds of users + the link facility has a hypertext-like user interface		+ form based user interfaces are well-supported - inadequate support for link following in the user interface
Performance			+ satisfactory performance

Figure 3.1. Differences between the major strengths and weaknesses of the data models, as discussed in the literature, and the toolkits, as found in the case study.

4. Discussion

The following discussion concerns the possibility of developing toolkits having the strengths of all three data models. It begins by motivating such a unification and by emphasizing changeability as a major property of a unified toolkit. Providing changeability is central to the relational model while it has only had a minor impact on the design of the two other data models. Mainly for this reason, it is the relational model which is considered for use as the nucleus of a unified TSARS development environment.

4.1 A unified toolkit approach

The text model, the hypertext model, and the relational model have different origins—library automation, human-computer interaction, and database theory, respectively. However, in recent years their application areas have come to overlap significantly. Thus, while some application areas are supported better by systems based on one of the data models, more and more applications seem to require facilities from two or all three classes of toolkits. Electronic publishing is one example: To provide online retrieval services, such a system requires some of

the text related query facilities of the text model and some of the browsing facilities of the hypertext model; and to enable extraction of text for inclusion in various, possibly overlapping, publications, the system also needs the data modelling and structured retrieval facilities of the relational model.

The advantages of using toolkits in software development rest on the assumption that the toolkit fits the application. Thus, a situation where facilities from more than one toolkit seem to be needed is a critical one. Furthermore, what appears to be needed when the choice of toolkit is made will inevitably be subject to subsequent modifications. The importance of this aspect is emphasized by it not being uncommon to see successful TSARS exist for 15-20 years. During their lifetime these systems are subject to manifold changes, and their continued success is largely due to the changes being incorporated into the original structure and idea of the system in a smooth way, see Naur (1985). This places high demands on the changeability of the toolkits used. The applications must, at the same time, be adaptable to manifold changes and manage to preserve their basic structure during this evolution. Thus, the toolkits must provide a flexible, yet stable, platform. Among other things, the stability should enable applications to benefit from achievements incorporated into new versions of the toolkit without more or less rewriting the systems. On the other hand, the flexibility should enable the systems to meet application-specific demands for tailoring and evolution.

Nishimoto & Ura (1989) note that in systems development response time and space requirements are mostly favoured at the expense of changeability. Probably, this reflects both an underestimation of the need for changeability and a pragmatic tendency to solve immediate problems before addressing longer term problems. This emphasizes that to reach a proper balance between performance and changeability both must be inherent in the toolkit—providing good performance without changeability is common and results in inflexible systems, providing changeability without good performance is research only.

4.2 Extending the relational model

The relational model focuses on one part of the application and is intended to be supplemented with tools handling, among other things, the user interface. Thus, relational DBMS have the openness required to form part of a TSARS development environment. Subject to meeting certain challenges, discussed below, the relational model is found capable of incorporating both the text model and the hypertext model:

- Incorporating the text model. As one of the examples in section 3 shows, the relational model can be extended with an inverted file, Boolean retrieval, and other specific text related facilities. In the example, Boolean retrieval is achieved by extending SQL with a special text retrieval clause; alternatively, Macleod (1979) shows how it can be implemented by adding a macro facility. Thus, provided the relational model is enhanced slightly, it seems suited for text model systems.
- Incorporating the hypertext model. The file and data structures used to implement the hypertext model are mostly special purpose ones, specific to the toolkit. However, larger applications place higher demands on the file and data structures and, partly for this reason, some hypertext toolkits are built on top of DBMS. Relational DBMS seem suited for this purpose, as they have the facilities for modelling all sorts of links. These links may refer to texts stored inside the database or in files external to it; thus, both closed systems and link services are supported.

There seems to be three major challenges involved in extending the relational model into a viable TSARS development environment incorporating the text model and the hypertext model: First,

the development of user interface tools for TSARS. Ready-made user interfaces should be provided for common applications. These user interfaces should be templates which may be used without modifications or refined to suit application-specific needs. As TSARS have a broad range of application, see figure 2.1, there is also a need for tools from which to build user interfaces, for example facilities creating relationships between the database and objects in the user interface. The user interface tools should support the development of TSARS combining facilities from the three data models. Currently, little guidance is available on how to combine for instance browsing and querying, but the importance of the subject is widely acknowledged, see for example Halasz (1988) and Marchionini & Shneiderman (1988).

Second, the need to access the texts at different levels of detail at different times. Many text model systems allow retrieval at two levels—document level and paragraph level, the paragraphs being defined by tags inserted into the texts. To allow retrieval of individual paragraphs, the relational model requires that the texts are divided into paragraphs, but then entire documents are necessarily retrieved as sequences of paragraphs. The need to access texts or other objects at several levels is the motivation for suggesting nested relations, see for example Jaeschke & Schek (1982) and Roth et al. (1988). However, a much simpler solution would be to allow concatenation of multiple paragraphs at retrieval time, much in line with for example the *sum* aggregate function.

Third, the performance costs of the high level of changeability. The response times of relational DBMS appear to be acceptable at least for small and medium sized TSARS, i.e. text databases less than 100 Mb of text. However, experimental evidence is scarce, especially concerning large and very large text databases. Many resources are invested in improving response times. Typically, these efforts focus on reducing the number of joins by abandoning the first normal form or on improving the query optimizer, for example through preprocessing and lazy evaluation, see for instance Graefe (1993) and Lynch (1991). A more narrowly focused effort could be to tune the query optimizer especially for text retrieval.

The space requirement is large; it is comparable to the 50-300% storage overhead seen in connection with the text model (Faloutsos, 1985). This is, partly, due to the duplication of keys needed to establish connections between the relations. Hertzum et al. (1993) report a storage overhead of 400% using a relational DBMS, but also find that this is of minor importance due to the rather low price of high volume storage media. Date (1986a) agrees that the space requirement is large in most current relational DBMS, because the relations are mapped into stored files. However, Date notes that due to the physical data independence a relational database could use any storage structure. This means that, in principle, the space requirement can be reduced to the same as any other system requires.

5. Conclusion

This study has reviewed the major strengths and weaknesses of three data models for the development of TSARS—the text model, the hypertext model, and the relational model. All three data models have unique, valuable properties, but more and more applications seem to require facilities from more than one of the data models. This requirement is recognized in the three commercially available toolkits described, and it is the motivation for the unified TSARS development environment discussed.

The unified toolkit is based on the relational model though this choice is contrary to most of the literature. The relational model was chosen because of its emphasis on changeability and without implying that the two other data models are unsuited. In general, the relational model seems capable of providing a flexible, yet stable, platform incorporating the text model and the

hypertext model. Specifically, the study points to three areas where relational DBMS should be improved to function as the basis of an efficient TSARS development environment: (1) Relational DBMS should be supplemented with user interface tools specifically for TSARS. On this point, much inspiration and experience can be gained from hypertext systems due to their emphasis on human-computer interaction. (2) Relational DBMS should be extended to provide access to the texts at several, application defined levels of detail. It seems possible to achieve this with simple means at retrieval time. (3) Performance, especially response times, should be improved to reduce the costs of the high level of changeability. However, though response times are important, they must be balanced with the decrease in overall performance caused by lack of changeability, for example prolonged system down time during maintenance and evolution.

Acknowledgments

I am grateful to Erik Frøkjær for stirring and spurring my interest in using the relational model as the basis for TSARS and to Kaj Grønbæk and Jørgen Lindskov Knudsen for volunteering their informed opinions on the status and directions of hypertext.

References

Andersen, K. H., Davidsen, P., Foldbjerg, M., Göttsche, P., Hertzum, M., Jensen, J., Jensen, L. G., Lund, K., Rehn, M. & Rungø, P. (1992). *Undersøgelse af værktøjer til opbygning af tekstsøgesystemer*. student report 92-4-12. DIKU, Copenhagen. In Danish.

Blair, D. C. (1988). An extended relational document retrieval model. *Information Processing & Management*, **24**(3), 349-371.

Borgman, C. L. (1986). Why are online catalogs hard to use? Lessons learned from information-retrieval studies. *Journal of the American Society for Information Science*, **37**(6), 387-400.

Bush, V. (1945). As we may think. *Atlantic Monthly*, **176**(1), 101-108.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, **13**(6), 377-387.

Codd, E. F. (1982). Relational database: a practical foundation for productivity. *Communications of the ACM*, **25**(2), 109-117.

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

Date, C. J. (1986a). Some relational myths exploded. In *Relational Database: Selected Writings*. pp. 77-123. Addison-Wesley, Reading, Massachusetts.

Date, C. J. (1986b). A practical approach to database design. In *Relational Database: Selected Writings*. pp. 417-470. Addison-Wesley, Reading, Massachusetts.

Engelbart, D. C. (1963). A conceptual framework for the augmentation of man's intellect. In *Vistas in Information Handling*. Vol. 1. pp. 1-29. Spartan Books, London.

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

Graefe, G. (1993). Query evaluation techniques for large databases. *ACM Computing Surveys*, **25**(2), 73-170.

Halasz, F. (1988). Reflections on Notecards: seven issues for the next generation of hypermedia systems. *Communications of the ACM*, **31**(7), 836-852.

Halasz, F. & Schwartz, M. (1990). The Dexter hypertext reference model. In *Proceedings of the Hypertext Standardization Workshop*. pp. 95-133. NIST Special Publication 500-178. National Institute of Standards and Technology, Gaithersburg, Maryland.

Hertzum, M., Søes, H. & Frøkjær, E. (1993). Information retrieval systems for professionals: a case study of computer supported legal research. *European Journal of Information Systems*, **2**(4), 296-303.

Jaeschke, G. & Schek, H.-J. (1982). Remarks on the algebra of non first normal form relations. In *Principles of database systems. Proceedings of the ACM SIGACT-SIGMOD Symposium* (Los Angeles, California). pp. 124-138.

Lynch, C. A. (1991). Nonmaterialized relations and the support of information retrieval applications by relational database systems. *Journal of the American Society for Information Science*, **42**(6), 389-396.

Lynch, C. A. & Stonebraker, M. (1988). Extended user-defined indexing with application to textual databases. In *Proceedings of the fourteenth international conference on very large databases, VLDB*. pp. 306-317. Morgan Kaufman, Palo Alto, California.

Macleod, I. A. (1979). SEQUEL as a language for document retrieval. *Journal of the American Society for Information Science*, **30**, 243-249.

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

Marchionini, G. & Shneiderman, B. (1988). Finding facts vs. browsing knowledge in hypertext systems. *IEEE Computer*, **21**(1), 70-80.

Morissey, J. M., Harper, D. J. & van Rijsbergen, C. J. (1986). Interactive querying techniques for an office filing facility. *Information Processing & Management*, **22**(2), 121-134.

Naur, P. (1985). Programming as theory building. *Microprocessing and Microprogramming*, **15**(5), 253-261.

Nelson, T. H. (1967). Getting it out of our system. In *Information Retrieval: A Critical Review*. Thompson Books, Washington, D. C.

Nishimoto, H. & Ura, S. (1989). Complex view support for a library database system. *Information Processing & Management*, **25**(5), 515-525.

Parsaye, K., Chignell, M., Khoshafian, S. & Wong, H. (1989). *Intelligent databases. Object-oriented, deductive hypermedia technologies*. Wiley & Sons, New York.

Puttress, J. J. & Guimaraes, N. M. (1990). The toolkit approach to hypermedia. In *Hypertext: concepts, systems and applications. Proceedings of the First European Conference on Hypertext*. pp. 25-37. Cambridge University Press, Cambridge.

Roth, M. A., Korth, H. F. & Silberschatz, A. (1988). Extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems*, **13**(4), 389-417.

Salton, G. & McGill, M. J. (1983). *Introduction to modern information retrieval*. McGraw-Hill, New York.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Second edition. Butterworths, London.