Strengths and Weaknesses of Database Models for Textual Documents

B.N. Rossiter† and M.A. Heather‡

† *Computing Laboratory, Newcastle University, Newcastle upon Tyne, England NE1 7RU*
‡ *Sutherland Building, Newcastle Polytechnic, Newcastle upon Tyne, England NE1 8ST*

ABSTRACT: User requirements in large and complex textbases are discussed in the light of current models. Examples applying relational and semantic models suggest criteria for a more fundamental approach involving the merger of object-oriented programming techniques with database methods in future complex object textbases.

KEYWORDS: document modelling, databases, complex objects.

1 Introduction

Little attention has been paid to text as structured data. Much of administrative data is in the form of textual strings but these tend to be treated as atomic entities independent of any relationship between words. Text retrieval and hypertext systems are based on physical divisions in documents and physical positions of words and rely on features like inversion, position operators and physical connections. By exploiting fully current technology such as multi-windowing and the emerging object-oriented programming, there have been significant advances in document manipulation in the provision of natural user interfaces [Pasquier-Boltuck et al 1988] and browsing systems [Brown 1988; Furuta and Stotts 1989]. There has been little regard for the very fine logical structure that lies beneath the physical form, even in recent data models for hypertext [Tompa 1989]. To handle large amounts of data of complex structure, more advanced file handling techniques will be required in the areas of full text information systems, electronic publishing, email, office automation, bulletin boards and conferencing.

Database technology needs to be extended from its present emphasis on simple objects to deal with complex objects such as text [Stonebraker et al 1987; Heather and Rossiter (in press)], CAD/CAM, CASE, knowledge

bases and complex business information. In this paper, the work reported at WOODMAN'89 [Heather and Rossiter 1989] is refined with particular attention being paid to the semantics of class structures and symbolic keys. The aim is to show the potential of the developing object-oriented database technology for modelling textual documents.

2 Demands of Textual Applications on Filing Systems

As a first step in formulating a model, the demands made by textual applications on the technology of filing systems have been analyzed by drawing on applications at both Newcastle and elsewhere. The first two columns of figure 1 summarize the results which have been reported more fully elsewhere [Rossiter and Heather 1990]. This paper considers the implications of the requirements for database systems. Besides the obvious structural properties listed in section 1 of figure 1, many of the other needs also place demands on textual filing systems: context and proximity matching require fine index structures; thesauri introduce further types of data; referential transparency, to provide navigational facilities as in hypertext, requires links from one text to another ideally using symbolic identifiers; trails made while navigating text structures need to be recorded as fully-fledged data [Zellweger 1989; Sillitoe et al 1990]; and updating involves the major problems of version management and the modelling of dynamic behaviour such as the life cycle of a document.

Three inherent requirements for modelling text structures need to be satisfied for a successful database initiative: dynamic selection of unit size; representation of non-hierarchical text structures; and description of data in generalized and specialized forms.

Unlike informal systems which might store text as a continuous stream of characters, formal systems need an object unit. Traditionally a choice has been made governed by the storage capability of the system, by the human capacity for searching, retrieving and comprehending the information, and by the character of the document which will often be determined by traditional printing techniques. Thus, for example, the size of a legal statute is controlled by parliamentary business and other political factors. However, one unit alone is insufficient for all purposes. With text objects, it should be possible to retain the ability of natural language to keep the choice of unit dynamic and with the option of lazy evaluation to postpone any decision until the full circumstances and context of use are known.

	Free Text	Relational		Semantic		Object-oriented
		stand.	extend.	E-R extend.	Taxis	
1. Design of STRUCTURE for holding text						
. unlimited size of fields and records	yes	not yet	not yet	-	-	-
. symbolic identification of records	limited	yes	yes	yes	yes	yes
. data models						
. hierarchical	yes	yes	yes	yes	yes	yes
. non-hierarchical (shared sub-obj.)	no	no	diff.	yes	yes	yes
. ability to retain un-normalized data	yes	no	no	-	yes	yes
. dynamic control of unit size (aggreg)	no	diff.	diff.	no	yes	diff.
. generalization and specialization (inher.)	no	no	no	yes	yes	yes
2. VIEWS						
. derived structures	no	yes	yes	no	yes	yes
. parallel texts	no	no	poss.	-	poss.	poss.
3. RETRIEVAL						
. fast	yes	no	yes	-	-	-
. non-procedural interactive languages	yes	yes	yes	-	no	no
. closure	no	yes	yes	-	no	no
. words + phrases in text						
. context	yes	yes	yes	-	yes	yes
. proximity matching	yes	no	yes	-	yes	yes
. keywords						
. free vocabulary	yes	yes	yes	-	yes	yes
. controlled vocabulary (thesauri, stop)	yes	yes	yes	-	yes	yes
. 'formatted' data	limited	yes	yes	-	yes	yes
. identifiers of text (symbolic key)	limited	yes	yes	-	yes	yes
4. Various formats for DISPLAY						
. human	yes	yes	yes	-	-	-
. machine-machine (wp, mark-up)	yes	yes	yes	-	-	-
5. TEMPORAL management with consistent updating						
. in-place modification, addition of data,	yes	yes	yes	-	yes	yes
. dynamic behaviour - control of doc. life	no	no	no	no	yes	yes
. version management	no	no	no	no	yes	yes
. concurrent access	yes	yes	yes	-	-	diff.
. value inheritance for natural data loading	no	no	no	yes	yes	yes
6. INTEGRITY						
. protection against hardware failures	yes	yes	yes	-	-	-
. referential	no	yes	yes	no	yes	yes
. value	yes	yes	yes	-	yes	yes
7. SECURITY						
. whole file	yes	yes	yes	-	yes	yes
. designated fields, data driven	yes	yes	yes	-	yes	yes
8. NAVIGATION through texts following conceptual paths						
. referential transparency (hypertext)	no	yes	yes	-	yes	yes
. trail maintenance	no	yes	yes	-	yes	yes
9. Textual ANALYSIS						
. function integrated with data	no	no	no	no	yes	yes
. word frequency lists, distribut. freq.	yes	no	yes	no	yes	yes
. statistical tests e.g. sentence length	yes	no	yes	no	yes	yes
. word co-occurrences	yes	no	yes	no	yes	yes
10. SEMANTIC aids						
. parsing	no	no	poss.	-	poss.	poss.
. predicate logic, machine translation	no	no	poss.	-	poss.	poss.
. cognitive textual types	no	no	poss.	-	poss.	poss.
11. MULTI-MEDIA						
. integration of text and other data	no	no	poss.	poss.	poss.	poss.

Figure 1: User Requirements for Textbases and their Satisfaction by Database Techniques.

There are texts for which hierarchical structures are inadequate. Shakespeare and legal texts are good examples. Their essential characteristic is that units may need to be linked to multiple units at higher levels of the tree structure rather than the single unit allowed in hierarchical structures. Such structures suggest the need to examine models described later where words are considered as atoms of data to be built dynamically into a variety of complex molecular objects.

Linked particularly with the navigation requirements described earlier is the need to generalise when describing text structures. For example, in a hierarchical text structure, any one part of the tree may usually cite any other part. The textbase can be viewed at two levels: generalisation for an abstract overview in which any type of text object cites any other type; and specialization for a more detailed representation in which a specific type of text object cites another specific type.

3 Semantic Models and Text Structures

Database models can be categorized into two main types: basic and semantic. A range of semantic models has been proposed in order to incorporate more features, constraints and abstractions than are found in the basic ones in an attempt to represent more closely the real world. These include the Entity-Relationship (E-R) Model [Chen 1976] and Taxis [Mylopoulos et al 1980]. Text because of its complex nature usually requires full semantic models to capture completely its structure and examples of Chen, Taxis and others have been developed at Newcastle.

3.1 Models for Expressing Static Aspects

The viewpoint of Chen is that database design is concerned primarily with the occurrence of entities and the relationship between them. An E-R diagram of a UK statute could be represented in the form of figure 2(a) using rectangles to denote entity-types and diamond-shapes to denote relationships. A relationship flagged '*' is mandatory, otherwise it is optional. All relationships are one-to-many (1:N) bar one. The idea of generalisation is employed with the scope of a generic entity-type being delineated by the enclosure of its associated specializations within a thickly-lined rectangle. The generic structures defined are *node* to represent all possible text units from an act through parts and schedules to subsections and subparagraphs and *text* to represent the units holding the main part of the text - section,

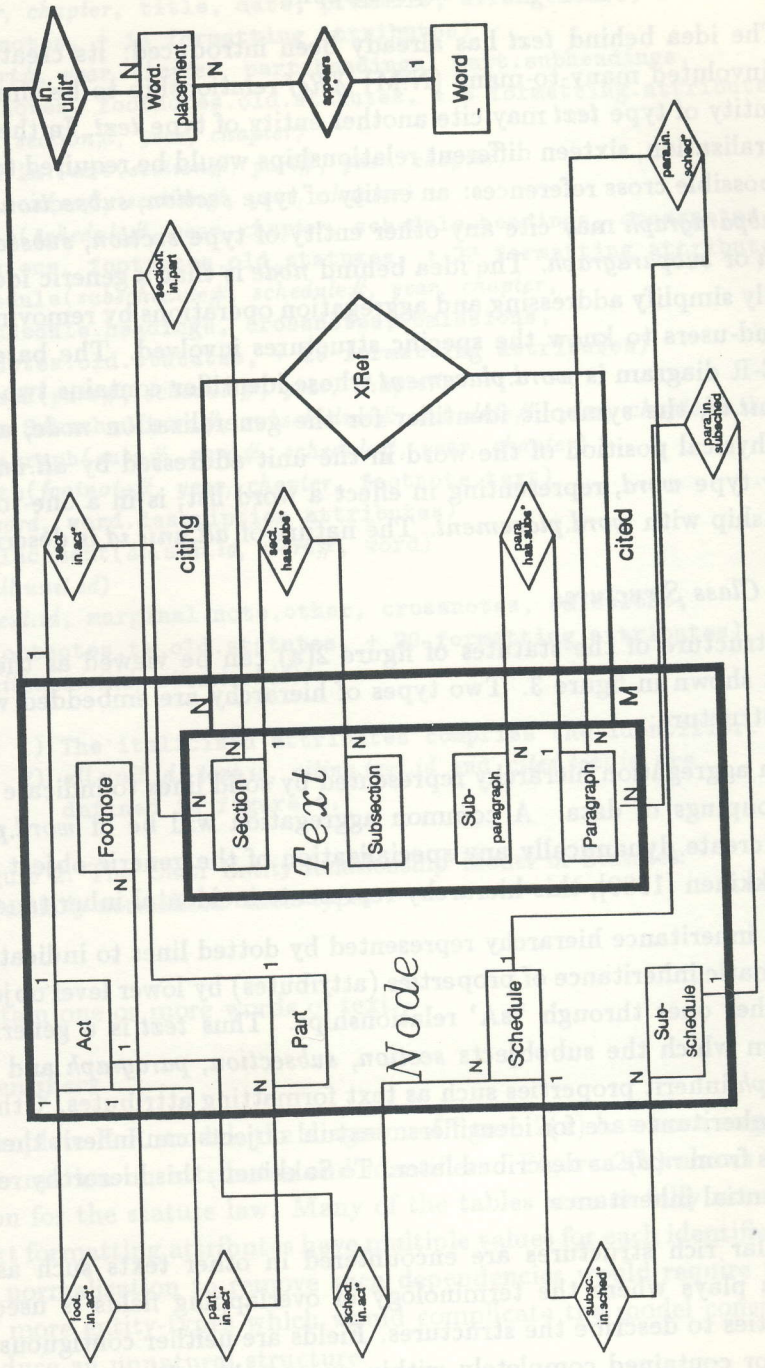

Figure 2: The Chen Entity-Relationship Model of Statutes: (a) Diagram

subsection, paragraph and subparagraph.

The idea behind *text* has already been introduced: its creation enables the involuted many-to-many (N:M) *XRef* relationship to be simplified into an entity of type *text* may cite another entity of type *text*. In the absence of generalization, sixteen different relationships would be required to handle all the possible cross references: an entity of type *section*, *subsection*, *paragraph* or *subparagraph* may cite any other entity of type *section*, *subsection*, *paragraph* or *subparagraph*. The idea behind *node* is that a generic identifier can greatly simplify addressing and aggregation operations by removing the need for end-users to know the specific structures involved. The base object in the E-R diagram is *word.placement* whose identifier contains two attributes *all.unit.id*, the symbolic identifier for the generalization *node*, and *word#* the physical position of the word in the unit addressed by *all.unit.id*. The entity-type *word*, representing in effect a word list, is in a one-to-many relationship with *word.placement*. The nature of *all.unit.id* is described later.

3.2 Class Structures

The structure of the statutes of figure 2(a) can be viewed as the complex object shown in figure 3. Two types of hierarchy are embedded within the class structure:

- An aggregation hierarchy represented by solid lines to indicate potential groupings of data. A common aggregation will be of *word.placement* to create dynamically any specialization of the generic object *node*. To Sakkinen [1989], this hierarchy represents incidental inheritance.
- An inheritance hierarchy represented by dotted lines to indicate the automatic inheritance of properties (attributes) by lower level objects from higher ones through 'isA' relationships. Thus *text* is a generic object from which the subobjects *section*, *subsection*, *paragraph* and *subparagraph* inherit properties such as text formatting attributes. Other forms of inheritance are for identifiers: textual objects can inherit their identifiers from *node* as described later. To Sakkinen, this hierarchy represents essential inheritance.

Similar rich structures are encountered in other texts such as Shakespeare's plays where the terminology of overlapping fields is used in the humanities to describe the structures. Fields are neither contiguous to each other nor contained completely within one another: lines, stage directions and speeches overlap each other with no clear structure other than that they

```
Act(year, chapter, title, date, preamble, arrangements,
  crossnotes, + 14 formatting attributes)
Part(part#, year, chapter, part.headings, part.subheadings,
  crossnotes, footnotes.old.statutes, + 5 formatting.attributes)
Section(section#, year, chapter)
Section.in.Part(section#, part#, year, chapter)
Subsection(ss#, section#, year, chapter)
Schedule(schedule#, year, chapter, schedule.headings, crossnotes,
  omissions, footnotes.old.statutes, + 29 formatting attributes)
Subschedule(subschedule#, schedule#, year, chapter,
  subschedule.headings, crossnotes, omissions,
  footnotes.old.statutes, + 29 formatting attributes)
Paragraph(para#, schedule#, year, chapter)
Para.in.Subsched(para#, subschedule#, schedule#, year, chapter)
Subparagraph(subp#, para#, schedule#, year, chapter)
Footnote(footnote#, year, chapter, footnote.text)
Word(word, word.description attributes)
Word.placement(all.unit.id, word#, word)
Node(all.unit.id)
Text(text.id, marginal.note.other, crossnotes, omissions,
     footnotes.to.old.statutes, + 20 formatting attributes)
XRef(citing.text.id, cited.text.id)

Notes: 1) The italicised attributes comprise the identifier.
       2) all.unit.id, text.id, citing.text.id and cited.text.id are
          defined in figure 4.
```

Figure 2: The Chen Entity-Relationship Model of Statutes:
(b) Partially-normalized Table-types

each contain one or more words of text.

3.3 Identifiers

For a complete E-R model, the diagram of figure 2(a) has been augmented with information on attributes and identifiers. Figure 2(b) shows this information for the statute law. Many of the tables are not fully normalised: some text formatting attributes have multiple values for each identifier value. Further normalisation to remove such dependencies would require the creation of more entity-types which would complicate the model considerably and produce an unnatural structure.

Figure 4 defines the generic identifiers in Taxis-like class structures. The

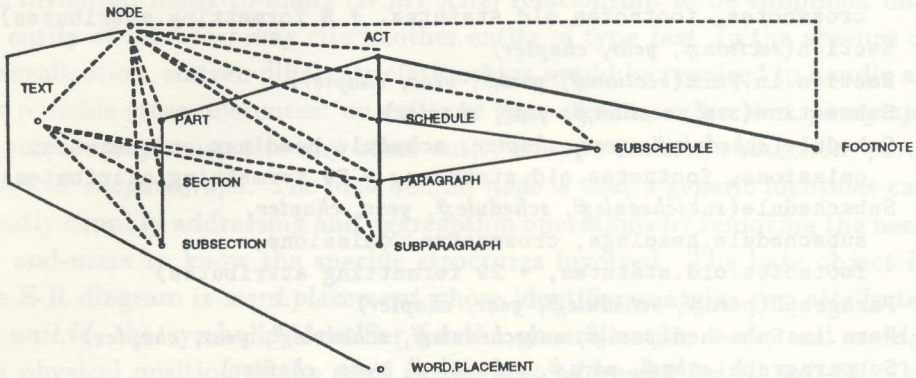

Figure 3: Class Structure for Objects Occurring in Legal Text

```
define AnyDataClass Node
  ss#:{|ssmin:ssmax|}
  section#:{|sectmin:sectmax|}
  part#:{|partmin:partmax|}
  subp#:{|subpmin:subpmax|}
  para#:{|paramin:paramax|}
  subschedule#:{|subsmin:subsmax|}
  schedule#:{|schmin:schmax|}
  footnote#:{|footmin:footmax|}
  year:{|yearmin:yearmax|}
  chapter:{|chapmin:chapmax|}
 unique
  all.unit.id:(year,chapter,part#,
   section#,ss#,schedule#,
   subschedule#,para#,subp#,
   footnote#)
```

```
define AnyDataClass Text isA Node
 changeable
  marginal.note.other:string
  crossnotes:string
  omissions:string
  footnotes.old.statutes:string
  formatting.attribute1:string
  formatting.attribute2:string..
 unique
 text.id:(year,chapter,section#,
  ss#,schedule#,para#,subp#)
```

```
define AnyDataClass XRef
  citing.text.id:set of Text
  cited.text.id:set of Text
 unique
  XRef:(citing.text.id,
        cited.text.id)
```

Figure 4: Taxis-like Specification of Symbolic Key for Statutes

identifier *all.unit.id* of the class *node* is a polynomial comprising the hierarchical sequence of an object in the statute law structure. In this study, the polynomial components are integers within the ranges shown, thus *ss#* takes

values between *ssmin* and *ssmax*. The identifier of the other generic entity-type *text* is defined as *text.id* with a subset of the attributes of *all.unit.id* inherited in the 'isA' relationship between *node* and *text*. The attributes of the entity *XRef* are the keys of the citing and cited text units *citing.text.id* and *cited.text.id* respectively. These attributes draw their values from the domain *text.id*. Generic symbolic keys have been used throughout our work to provide a powerful mechanism for flexibly manipulating the complex object structure of figure 3 [Rossiter and Heather 1988].

3.4 Models for Expressing Dynamic Aspects

Major deficiencies of the E-R and Borkin models are that they have no defined operations and thus cannot handle dynamic control of the life-cycle of entities. Semantic models which enable such dynamic structures to be expressed as well as static ones have therefore also been examined such as Taxis, the Event Model and SHM+. The model Taxis illustrates the potential of this approach and its use is being explored at Newcastle for control of the legal drafting process [Rossiter and Heather 1990].

4 Basic Models and Text Structures

4.1 The Relational Model

None of the basic models is rich enough in capability to satisfy all requirements in manipulating complex textual structures. Hierarchical and network models can be quickly discounted but a more detailed discussion is necessary to illustrate weaknesses in a relational approach 'flattening' the data. The relational model offers the power of the network model but with a simple and elegant method of data manipulation. The E-R model given earlier for law can be implemented directly by mapping the table-types in figure 2(b) on to conventional relations: one table-type per relation. There is a major difficulty in this transformation for text structures: the unnormalized data for figure 2(b) cannot be retained in the relational model. Also, the semantics of the E-R diagram are now represented implicitly; for instance, the structure of the objects:

act -> schedule -> paragraph -> subparagraph -> word.placement

is represented by a series of relations whose attributes carry the inter-object relationships so that the basic hierarchical structure is not explicitly conveyed to the user as in the E-R diagram. For users with detailed knowledge

of the relational model, a clear advantage is the ease with which keys as formatted data can be manipulated giving flexibility in unit size for display and symbolic addressing for navigation. Standard set operators can provide aggregation of data, dynamic variation of the unit of retrieval as users' needs change, cross-referencing and closure. Users with little knowledge of the relational model, however, are dependent on views predefined by the database administrator to achieve abstractions such as aggregation.

There is also the problem of indexing text in a versatile manner. There is no concept of a global index: a single index cannot readily be constructed for a series of attributes in different relations across the textbase to facilitate searching on a particular generalized abstraction. It is not easy to build a single word index on the abstraction *text* defined earlier.

4.2 Extended Relational Approach

The logical approach for improving the flexibility of relational systems in manipulating text is to flatten the data so that it is completely normalized to the word level. The existence of the base relations shown in figure 2(b) and of relations holding word positions in the text should provide data structures which can satisfy most user requirements. However, some features of figure 2(b) are not easy to represent. The keys *text.id*, *citing.text.id* and *cited.text.id* essential for generalization cannot strictly be defined as in figure 4 for that conflicts with the entity integrity rule: no component of a primary key can be null. This can be overcome by recording null values as zeroes. For the purists, though, three logical word indexes are needed in the form of the relations shown below to represent the three distinct paths to *word.placement* via *footnote*, *subsection* or *subparagraph*. Also shown is the relation *request* which contains the words in the user's search request.

Word.H2 (*word#, footnote#, year, chapter*, word)
Word.H3 (*word#, ss#, section#, year, chapter*, word)
Word.H4 (*word#, subp#, para#, schedule#, year, chapter*, word)
Request (*word*)

The word index relations can be quickly searched by dividing them by the relation *request*. The unit searched can be varied dynamically by taking appropriate views of the index using the projection operator. Thus if relation *request* contains the search terms A and B, a search for A and B in the same subsection can be achieved by:

```
Word.H3 [year, chapter, section#, ss#, word] divideby Request[word]
```

and in the same schedule by:

```
Word.H4 [year, chapter, schedule#, word] divideby Request[word]
```

A clear disadvantage of the approach of flattening the data is that no standard set operators exist for constructing the word indexes and it is not realistic to expect users to input such structures manually. Operators to perform the flattening of normal text could be user-written but this would involve additional effort and would be likely to result in inefficient code.

The manipulation of biblical text in relational databases has also been considered in detail [Heather and Rossiter (in press)]. It was found feasible to normalize this data to the word level and employ SQL or relational algebra for searching the data against different unit sizes and for aggregating the data as necessary. The biblical data are represented by a complex object with the single path:

testament -> book -> chapter -> verse -> word.placement

Although this path is much simpler to manage than the multiple paths of law shown above, the approach of 'flattening' the data was found to be cumbersome for data manipulation, to hide the natural structure of the data from the user and to have adverse performance implications when reconstituting aggregations for documents in large textbases. Only with extensive mediation, between the system and the user, is a natural interface provided with a high level of data abstraction.

5 Object Oriented Systems

In advanced languages such as Ada, C++ and Simula, the concept of class structure and variable unit size is well established through the extensible type system with the ability to declare abstract data types. Some of these languages allow subobjects to inherit properties from higher-level objects and inter-object communication. These object-oriented systems readily allow iterative searching of complex objects, multiple levels of abstraction and a natural ability to handle dynamic aspects with function fully integrated with data [Bloom and Zdonik 1987], all important issues for textbases.

The use of object-oriented programs for database management is in its early stages. Advances depend on programming systems being developed to handle persistent data such as in the early work by Atkinson with PS-Algol [Atkinson et al 1981]. One of the first developments was GemStone

[Copeland and Maier 1984] which is related to SmallTalk and uses the Opal language for data definition and manipulation. Abstract data types can be defined, object identity is preserved and objects participate in one or more collections to provide a shared subobject facility. Behavioural aspects are handled by messaging.

The strengths of the object-oriented approach lie in the ability to import advanced programming techniques into areas of data modelling in which database technology has been traditionally weak. However, in the management of persistent data, object-oriented systems have a number of significant weaknesses. These include many of the standard functions which are an essential part of any database system. Thus security, concurrency, transaction control, archiving and some aspects of integrity are achieved by primitive methods, if at all. Optimisation of data storage and indexing are at an early stage perhaps analagous to that of the first relational systems.

Of greatest significance, perhaps, is that owing to their procedural nature, many object-oriented systems do not provide the non-procedural interactive languages that end-users require for data manipulation. Procedural interfaces requiring some knowledge of high-level programming languages may be acceptable in engineering applications where the clientale usually has a relatively sophisticated programming background. However, in areas such as text and office automation, it is considered that procedural interfaces are not appropriate to the environment. Clearly, ad hoc query languages could be designed for applications by writing an interface program in a host language. However, the more durable non-procedural languages have been based on mathematical methods, such as relational calculus and algebra, applied to a conceptual model of the data. There is thus, owing to a lack of emphasis on conceptual modelling techniques, a layer of control missing from current object-oriented systems to provide the necessary user environment. There are also difficulties with closure: if the result of a query is presented as a table, that is not a viable structure for further work.

6 Discussion

The last six columns of figure 1 show the extent to which our critical requirements for textbases are met by the techniques of free text retrieval, ISO-standard relational database, extended relational database with facilities to flatten textual data, semantic models oriented towards static and dynamic aspects such as E-R and Taxis respectively, and object-oriented

systems. A hyphen indicates incomplete information is available.

Free text retrieval systems suffer from limited data structuring ability, lack of navigational aids and an inability to model dynamic behaviour. Standard relational systems provide better data structuring and navigational facilities but their performance in context searching, other than on base units, is questionable and proximity searching is not available. Extended relational systems with flattened textual data can achieve a better performance and, through some ability to model complex objects, provide the basis of a unified model for multi-media data and of initiatives in advanced text processing such as semantic parsing. However, aggregation is a cumbersome task for a user and dynamic behaviour is not addressed. The E-R model has not been directly implemented so the information in the figure is incomplete but, with the lack of defined operations, it cannot be a complete solution.

Whether in the guise of semantic models like Taxis or databases such as GemStone, object oriented approaches appear to offer the most promise. Such systems handle quite naturally variable unit size, shared subobjects, dynamic behaviour and integration of function and data, and have considerable promise in multi-media modelling. The semantic models, in particular, also handle aggregation well through subtyping declarations. However, so far, object-oriented systems have presented relatively procedural interfaces to users, do not readily provide closure, are rather limited in standard database functions such as concurrent access and have not proved themselves in terms of performance. The optimum solution for users of textbases would therefore appear to be a merger of advanced database technology as in semantically-enriched relational systems with advanced object-oriented programming to create object-oriented textbases. Such textbases should be thought of as object-bases rather than pure database or object-oriented systems. It should not be pretended that such a merger will be easy. The cultural differences between the two approaches present many difficulties [Tsichritzis and Nierstrasz 1988] and much research of a fundamental nature is still required to attain a single complete multi-media model.

References

[Atkinson81] M P Atkinson, K J Chisholm, and W P Cockshott (July 1981), PS-Algol: an Algol with a persistent heap, *ACM SIGPLAN Notices* **17**(7).

[Bloom87] T Bloom and S B Zdonik (1987), Issues in the Design of Object-oriented Database Languages, OOPSLA'87 Conf. Proc., *ACM SIGPLAN Notices* **22**(12) 441-451.

[Brown88] P J Brown (1988), Hypertext: the way forward, in: *Document Manipulation and Typography,* ed. J C van Vliet, Cambridge 183-191.

[Chen76] P P-S Chen (1976), The Entity-Relationship Model – towards a unified view of data, *ACM TODS* **1**(1) 9-36.

[Copeland84] G Copeland and D Maier (1984), Making SmallTalk a Database System, *Proc ACM/ SIGMOD Int. Conf. Management of Data.*

[Furuta89] R Furuta and P D Stotts (1989), Programmable Browsing Semantics in Trellis, in: *Hypertext'89 Proc.,* Special Issue - SIGCHI Bulletin 27-42.

[Heather (in press)] M A Heather and B N Rossiter (in press), Syntactical Relations in Parallel Text, in: *Proc. 15th Int. ALLC Conf.,* ed. Y Choueka, Jerusalem 1988.

[Heather89] M A Heather and B N Rossiter (1989), Theoretical Structures for Object-based Text, in: *WOODMAN'89,* edd. J André and J Bézivin, BIGRE **63-64** 178-192.

[Mylopoulos80] J Mylopoulos, P A Bernstein, and H K T Wong (1980), A Language Facility for Designing Database-Intensive Facilities, *ACM TODS* **5** 185-207.

[Pasquier-Boltuck88] J Pasquier-Boltuck, E Grossman and G Collaud (1988), Prototyping an Interactive Electronic Book System using an Object-Oriented Approach, in: *Lecture Notes Comp. Sci.,* Springer-Verlag **322** 177-190.

[Rossiter88] B N Rossiter and M A Heather (1988), Data Models and Legal Text, *CC-AI* **5**(1) 39-55.

[Rossiter90] B N Rossiter and M A Heather (1990), *Towards the Object Oriented Text Base,* Computing Laboratory, University of Newcastle upon Tyne, Technical Report, no. 297.

[Sakkinen89] M Sakkinen (1989), Disciplined Inheritance, *ECOOP'89 Proceedings,* (ed.) Cook, S, Cambridge 39-56.

[Sillitoe90] T J Sillitoe, B N Rossiter and M A Heather (1990), Trail Management in Hypertext: in: *BNCOD-8 Proc.,* ed. A Brown, Pitman.

[Stonebraker87] M Stonebraker, J Anton, and E Hanson (1987), Extending a Database System with Procedures, *ACM TODS* **12**(3) 350-376.

[Tompa89] F W Tompa (1989), A Data Model for Flexible Hypertext Database Systems, *ACM Trans Information Systems* **7**(1) 85-106.

[Tsichritzis88] D C Tsichritzis and O M Nierstrasz (1988), Fitting Round Objects into Square Databases, ECOOP'88 Proceedings, in: *Lecture Notes Comp. Sci.,* Springer-Verlag **322** 283-299.

[Zellweger89] P T Zellweger (1989), Scripted Documents: A Hypermedia Path Mechanism, in: *Hypertext'89 Proc.,* Special Issue - SIGCHI Bulletin 1-14.