University of Newcastle upon Tyne

COMPUTING LABORATORY

Towards the Object-Oriented Textbase

B.N. Rossiter and M.A. Heather

TECHNICAL REPORT SERIES

No 297 November, 1989

TECHNICAL REPORT SERIES
No. 297 November, 1989

Toward the Object-Oriented Textbase

B.N. Rossiter and M.A. Heather

Abstract

Text is an important component in most types of information systems. Between "free text" methods with no regard to content and fully intelligent natural language processing lies a range of computational methods for handling the structure of text. These rely on semantic data typing and constructing machine models of document architectures and draw on database technology and the object-oriented approach. The object-oriented textbase aims to provide a best possible computer representation of the unreduced information communicated and stored in the medium of text.

Series Editor: M.J. Elphick

Printed and published by the University of Newcastle upon Tyne,
Computing Laboratory, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

ROSSITER, Brian Nicholas

Towards the object-oriented textbase. [By] B.N. Rossiter and M.A. Heater

Newcastle upon Tyne: University of Newcastle upon Tyne: Computing Laboratory, 1989.

(University of Newcastle upon Tyne, Computing Laboratory,
Technical Report Series, no. 297)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.
Computing Laboratory. Technical Report Series. 297

HEATHER, Michael A.

Abstract

Text is an important component in most types of information systems. Between "free text" methods with no regard to content and fully intelligent natural language processing lies a range of computational methods for handling the structure of text. These rely on semantic data typing and constructing machine models of document architectures and draw on database technology and the object-oriented approach. The object-oriented textbase aims to provide a best possible computer representation of the unreduced information communicated and stored in the medium of text.

About the author

B.N. Rossiter is a lecturer in the Computing Laboratory in databases and systems analysis.

M.A. Heather is senior lecturer in law at Newcastle Polytechnic where he has been responsible for computers and law since 1979.

Suggested keywords

COMPLEX OBJECTS
DATABASE SYSTEMS
OBJECT-ORIENTED DATABASES
RELATIONAL MODELS
SEMANTIC MODELS
TEXT MANAGEMENT

Suggested classmarks (primary classmark underlined)

Dewey (18th):	001.6442	348.02	029.7
U.D.C.	681.322.06	340.13	651.838.8

Towards the Object-Oriented Textbase

B N Rossiter
Computing Laboratory
Newcastle University
Newcastle upon Tyne
England NE1 7RU
email: B.N.Rossiter@newcastle.ac.uk

M A Heather
Sutherland Building
Newcastle Polytechnic
Newcastle upon Tyne
England NE1 8ST

1: Abstract

Text is an important component in most types of information systems. Between "free text" methods with no regard to content and fully intelligent natural language processing lies a range of computational methods for handling the structure of text. These rely on semantic data typing and constructing machine models of document architectures and draw on database technology and the object-oriented approach. The object-oriented textbase aims to provide a best possible computer representation of the unreduced information communicated and stored in the medium of text.

2: 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 packages and hypertext systems are built on physical divisions in documents and the physical position of words and can make use of features like inversion, position operators and physical connections. There is little regard for the logical structure that lies beneath the physical form. A need for more advanced file handling techniques has been brought about by the large quantities of electronic text which are being generated in full text information systems, electronic publishing, e-mail, office automation, bulletin boards and conferencing.

Advanced filing techniques as found in database systems can be applied to a wide class of data but current database technology has been developed for the simple data objects found in the narrow category of administrative data. There is a broad class of other data, sometimes called general data, that cannot be readily normalized even though it may contain a high degree of structure. Recently, it has been recognized by various writers that database technology needs to be extended from its present emphasis on simple objects for it to deal with general data involving complex objects such as text itself [Stonebraker, Anton, & Hanson 1987], [Heather & Rossiter 1989a], CAD/CAM [Lorie et al 1985], [Kemper & Lockemann 1987], CASE [Earl et al 1986] and knowledge bases [Zaniolo et al 1986]. In designing a model to manipulate general data more effectively, it is important to observe the principles of software engineering. If there is a natural structure of information applicable to all types of general data, an effective model should allow for the design of a single integrated database for storage and manipulation of all data. Furthermore a test of a model is not just its formal completeness but how natural it is in modelling the real world and how successful it is in preserving real-world data without losing information.

As a first step to formulating a full data model for text, the demands made by textual applications on the technology of filing systems are reviewed here by drawing on applications at Newcastle and elsewhere.

3: Demands of Textual Applications on Filing Systems

The first major database application at Newcastle to manipulate full text was in the legal area. Law in its various forms of statutes, cases and litigation exhibits two major features of full text data - volume and complexity. The volume of statute law currently in force is of the order of 300Mb for England & Wales alone. In America and Europe, large bodies of statute law are also found so that the law merits international attention. Case reports are a valuable source of precedents in many countries and these are produced in printed form by a number of publishers.

Figure 1 shows an extract from the case report of Wings Ltd v. Ellis where there was a straightforward issue of whether an incorrect description in a travel brochure constituted a criminal offence. This case illustrates the kind of information that is relevant to this question and the various other legal sources that may have to be consulted in resolving it. In order to represent readily its complex structure, the case report employs an elaborate presentation of text with diverse formats and multiple character sets and sizes. To capture adequately the detailed structure, sophisticated modelling techniques are required and these will be explored later. This figure also shows the place for database techniques in electronic publishing and typography. The printed version presents a fixed view of the structure which has to be accepted by all readers although different organizations of the text data are appropriate for different users. The way to exploit the power of computers for logical re-organization is by holding the complete structure of each statute and case in database records. Then, for example, the display format can be readily changed to meet new circumstances: presentation of output to the human eye requires quite different forms from operation in a machine-machine mode, where the output is directly into a word-processor or mark-up language, or where one machine interrogates another for information [Connolly 1985].

Users of a law database require selective retrieval of information. Where this can be performed using relatively non-procedural languages, such as relational calculus (e.g. with SQL) or relational algebra, the user may state what is wanted and let the system decide for itself how the request should be met. Thus, very high-level languages should be available which are relatively easy to learn and apply by users with no special training in computing skills. Alternatively an additional 'user-friendly' interface perhaps employing near-natural language and statistical techniques to resolve ambiguities can be built upon high-level query languages [Newton 1981] to provide simple access to users with no training.

Searches will be made typically for words, word stems and phrases contained within the text. With searches for the co-occurrence of two or more words, context is important: words matched may be required to co-occur within a given physical/logical unit of text such as a paragraph. Physical distances between words may also be used as search criteria so that one word may be required to be within a certain proximity of another. Facilities provided by current information retrieval systems may facilitate the user's task in optimizing his search request. Thus the maintenance of temporary result stacks and an iterative searching capability are essential components of the user interface.

In full-text retrieval, an area of long standing controversy [Salton 1986] is the relative performance of systems employing manual and automatic indexing methods. In the former, keywords are added to documents as tags for retrieval; in the latter, the words of natural language occurring in the text are used as

Figure 1
Extract from Case Report "Wings Ltd v. Ellis"

Types of Text are indicated as follows:

unmarked	-	whole-text
enclosed in angular brackets	-	short-text
underlined with solid line	-	force-text
underlined with broken line	-	loose-text

[HOUSE OF LORDS]

WINGS LTD. . . . . RESPONDENT

AND

ELLIS . . . . APPELLANT

1984 June 25, 26; Oct. 25

Lord Hailsham of St. Marylebone L.C., Lord Keith of Kinkel, Lord Scarman, Lord Brandon of Oakbrook and Lord Templeman

Trade Description—Travel agents—Accommodation facilities and services—Company unwittingly publishing false statements in travel brochure—Discovery of error—Company informing sales staff and agents with instructions to inform customers of error—Whether company knowingly making false statement—Trade Descriptions Act 1968 (c. 29), s. 14(1)(a)

The defendant company, in the course of their business as holiday tour operators, published a travel brochure which contained descriptions and photographs of hotel accommodation. Unknown to anyone within the company at the time of publication, the brochure falsely described a hotel in Sri Lanka as air conditioned by use of the code letters "A.C." and a photograph wrongly purporting to be that of a room at the hotel which, in so far as it gave no signs of any outside ventilation such as overhead ceiling fans or mosquito nets, indicated that the room was air conditioned. The errors were not discovered until May 1981 after the brochures had been distributed to travel agents. A memorandum, dated 1 June 1981, was sent to all company staff instructing them to amend their brochures and instructing sales agents to inform travel agents and customers of the errors when holiday bookings were made by telephone. Customers who had already booked holidays at the hotel were also informed of the mistake by letter. On 13 January 1982, the complainant read an unamended brochure and booked a holiday at the hotel with the company through travel agents. The complainant, who could only be contacted through the travel agent, was never told of the errors in the brochure. On his return from Sri Lanka, he complained to the company and to a trading standards officer that the hotel was not air conditioned as described in the brochure. The company was convicted by justices on informations preferred against them by the trading standards officer alleging that on 13 January 1982, in the course of a trade or business, they made a statement which they knew to be false as to the nature of the accommodation at the hotel, namely the statement "A.C." in the description of the hotel in the brochure, contrary to section 14(1)(*a*)(ii) of the Trade Descriptions Act 1968[1]; and that they recklessly made a statement which was false as to the nature of the accommodation at the hotel, namely the photograph in the brochure, which was likely to be taken as an indication that the hotel bedrooms were air conditioned, contrary to section 14(1)(*b*)(ii) of the Act of 1968. On appeal by the company the Divisional Court of the Queen's Bench Division allowed the appeal and quashed the convictions.

The prosecutor appealed in respect of the quashing of the conviction under section 14(1)(*a*) of the Act and the following point of law was certified: whether a defendant may properly be convicted of an offence under section 14(1)(*a*) of the Trade Descriptions Act 1968 when he has no knowledge of the falsity of the statement at the time of the publication but knew of the falsity at the time when the statement was read by the complainant.

On the prosecutor's appeal:—

Held, allowing the appeal, that the defendant company had been rightly convicted of committing an offence under section 14(1)(*a*) of the Act of 1968 since (*per* Lord Keith of Kinkel, Lord Scarman and Lord Brandon of Oakbrook) a statement which was false was made by the company in the course of its business when it was read by the complainant, an interested member of the public doing business with the company upon the basis of the statement; that the offence was committed on that occasion because the company then knew it was false to state that the hotel accommodation was air conditioned and the fact that the company was unaware of the falsity of the statement when it was published was irrelevant; that if the company considered that it was innocent of fault, it was open to it to prove lack of fault under the statutory defences, but it did not do so (post, pp. 281F, 290E, 297C–D, 298F–G).

Reg. v. *Thomson Holidays Ltd.* [1974] Q.B. 592, C.A. applied.

Per Lord Hailsham of St. Marylebone L.C. The certified question should be answered in the affirmative but with a qualification by saying: "Yes, unless the defendant has raised a successful defence under section 24 of the Act and provided that the reading by the complainant was part of the chain of consequences intended and authorised by the defendant prior to its receipt by the complainant" (post, p. 290D–E).

Per Lord Brandon of Oakbrook. (i) The certified question was ineptly expressed and should be amended as follows: "Whether a defendant may properly be convicted of an offence under section 14(i)(*a*) of the Trade Descriptions Act 1968 when he has made a continuing false statement, which he did not know was false when he first made it, but which having come to know of its falsity at some later time, he has thereafter continued to make." As amended in that way, it should be answered with a simple "Yes." (post, p. 298E–F). (ii) On the footing that the certified question can be answered as it stands, it should be answered in the manner proposed by Lord Scarman (post, p. 298F–G).

Per Lord Templeman. The certified question should be answered in the affirmative. The accused must plead and prove the circumstances specified in section 24 before a defence of mistake can succeed (post, pp. 300G, 301A).

Per Lord Hailsham of St. Marylebone L.C., Lord Keith of Kinkel, Lord Scarman and Lord Brandon of Oakbrook. *Reg.* v. *Thomson Holidays Ltd.* [1974] Q.B. 592 was correctly decided save in so far as it purports to decide as a general proposition of law applicable to all cases that a statement is only made for the purposes of section 14 of the Act of 1968 when it is communicated to someone (post, pp. 285D–F, 290E, 296D–E, 298B–C).

Decision of the Divisional Court of the Queen's Bench Division [1984] 1 W.L.R. 731; [1984] 1 All E.R. 1046 reversed.

[1] Trade Descriptions Act 1968, s. 14(1): see post, pp. 282G—283A.

The following cases are referred to in their Lordships' opinions:

Coupe v. Guyett [1973] 1 W.L.R. 669; [1973] 2 All E.R. 1058, D.C.
Gammon (Hong Kong) Ltd. v. Attorney-General of Hong Kong [1985] A.C. 1; [1984] 3 W.L.R. 437; [1984] 2 All E.R. 503, P.C.
Jackson v. Horizon Holidays Ltd. [1975] 1 W.L.R. 1468; [1975] 3 All E.R. 92, C.A.
Jarvis v. Swans Tours Ltd. [1973] Q.B. 233; [1972] 3 W.L.R. 954; [1973] 1 All E.R. 71, C.A.
M.F.I. Warehouses Ltd. v. Nattrass [1973] 1 W.L.R. 307; [1973] 1 All E.R. 762, D.C.
Mousell Brothers Ltd. v. London and North-Western Railway Co. [1917] 2 K.B. 836, D.C.
Reg. v. Miller [1983] 2 A.C. 161; [1983] 2 W.L.R. 539; [1983] 1 All E.R. 978, H.L.(E.)
Reg. v. Thomson Holidays Ltd. [1974] Q.B. 592; [1974] 2 W.L.R. 371; [1974] 1 All E.R. 823, C.A.
Reg. v. Treacy [1971] A.C. 537; [1970] 3 W.L.R. 592; [1970] 3 All E.R. 205, C.A.; [1971] A.C. 537; [1971] 2 W.L.R. 112; [1971] 1 All E.R. 110, H.L.(E.)
Sherras v. De Rutzen [1895] 1 Q.B. 918, D.C.
Smedleys Ltd. v. Breed [1974] A.C. 839; [1974] 2 W.L.R. 575; [1974] 2 All E.R. 21, H.L.(E.)
Sunair Holidays Ltd. v. Dodd [1970] 1 W.L.R. 1037; [1970] 2 All E.R. 410, D.C.
Sweet v. Parsley [1970] A.C. 132; [1969] 2 W.L.R. 470; [1969] 1 All E.R. 347, H.L.(E.)
Tesco Supermarkets Ltd. v. Nattrass [1972] A.C. 153; [1971] 2 W.L.R. 1166; [1971] 2 All E.R. 127, H.L.(E.)

The following additional cases were cited in argument:

Birkenhead and District Co-operative Society Ltd. v. Roberts [1970] 1 W.L.R. 1497; [1970] 3 All E.R. 391, D.C.
British Airways Board v. Taylor [1976] 1 W.L.R. 13; [1976] 1 All E.R. 65, H.L.(E.)
Coppen v. Moore (No. 2) [1898] 2 Q.B. 306, D.C.
Derry v. Peek (1889) 14 App.Cas 337, H.L.(E.)
Fagan v. Metropolitan Police Commissioner [1969] 1 Q.B. 439; [1968] 3 W.L.R. 1120; [1968] 3 All E.R. 442, D.C.

APPEAL from the Divisional Court of the Queen's Bench Division.

This was an appeal, by leave of the House of Lords (Lord Diplock, Lord Brandon of Oakbrook and Lord Templeman) given on 1 March 1984, by the appellant, David Kenneth Ellis, the prosecutor, from the judgment dated 2 December 1983 of the Divisional Court of the Queen's Bench Division (Robert Goff L.J. and Mann J.) allowing an appeal by the respondents, Wings Ltd., quashing their convictions of two offences under section 14 of the Trade Descriptions Act 1968 by justices at Plymouth Magistrates' Court on 17 January 1983. The appeal related to the first of two offences of contravening section 14 of the Trade Descriptions Act 1968, namely, the conviction under section 14(1)(*a*) of the Act. The Divisional Court certified that the following point of law of general public importance was involved in their decision, namely:

"Whether a defendant may properly be convicted of an offence under section 14(1)(*a*) of the Trade Descriptions Act 1968 where he has no knowledge of the falsity of the statement at the time of its publication but knew of the falsity at the time when the statement was read by the complainant."

The facts are stated in their Lordships' opinions.

Anthony Scrivener Q.C. and *Nicholas Nardecchia* for the appellant. The appeal raises two issues relating to the true construction of section 14(1)(*a*) of the Trade Descriptions Act 1968. They relate respectively to (1) the expression "to make a statement," and (2) the words that immediately follow, "which he knows to be false." So far as making a statement is concerned, it was thought by prosecutors throughout the country that the meaning of this expression was well established by *Reg. v. Thomson Holidays Ltd.* [1974] Q.B. 592 as meaning that a statement was made for the purposes of section 14(1) when it was communicated to a customer or consumer and not at the time of publication. As regards the second question, there have been problems. The proper approach to this question is by way of the doctrine of vicarious liability.

Reliance is placed on the following propositions: (1) The offence in section 14(1)(*a*) of the Trade Descriptions Act 1968 is a "conduct" offence and not a "result" offence and *a statement is made* when it is communicated to someone in the course of a trade or business. Accordingly, the respondents made the false statements on 13 January 1982 when their servants or agents communicated them to the complainant.

(2) The respondents knew that the statements were false because: (a) the wording of section 14(1)(*a*) of the Act provides: (i) ". . . which he knows to be false" and does not provide "knowingly to make a statement . . .". It follows that only knowledge of the falsity of the statement is required, and (ii) the subsection relates to knowledge of a fact and not to an intent. (b) The justices found as a fact, and were entitled to find, that the respondents knew that the statements were false. In particular, the respondents had tried to correct the falsity of such statements prior to 13 January 1982 which showed such knowledge. Alternatively, (c) at the time when the statements were made the respondents' servants or agents knew that they were false and the respondents are vicariously liable for the conduct of their servants or agents acting in the course of their employment. In the further alternative, (d) if the offence is a "result" offence then between the publication and when the statements were communicated to the complainant it was proved that the respondents knew that the statements were false or that the respondents' servants or agents, to whom the respondents were vicariously liable, knew that the statements were false, or it is a continuing offence.

(3) On the facts the respondents were entitled to seek to establish the defences in section 24 of the Act, but did not attempt to do so. In particular, (a) if the commission of the offence was due to the *mistake* of a "director, manager, secretary or other similar officer" of the respondents, then the respondents could have sought to prove the defence in section 24(1)(*a*) and (*b*), or (b) if the commission of the

the tags. Manual systems are further sub-divided into those where the choice of keywords is tightly controlled by a thesaurus and those where the indexer has a free choice of terminology. The free index approach is usually enhanced by the use of stop lists which prevent common words being used as index entries. A secondary commercial source of information such as the INSPEC abstracts electronically published by the IEE includes both free and controlled keywords as well as full text. All these need to be handled by some generalized system.

3.1 Cross-referencing

Some parts of the legal text in Figure 1 make reference to other texts in the same or different documents. Thus, in paragraph C on page 273, subsequent paragraphs of the same case are cited in the manner:

(post, pp. 281F, 290E, 297C-D, 298F-G)

In addition to these intra-case citations, there are inter-case citations as itemized on page 274 which have been divided by the author of the case report into two categories: those referred to by the Lord Justices in their opinion, for example:

Coupe v. Guyett [1973] 1 W.L.R. 669; [1973] 2 All E.R. 1058, D.C.

and those used by the respondent and appellant in their arguments, but not referred to in the judgments, for example:

Coppen v. Moore (No. 2) [1898] 2 Q.B. 306, D.C.

There are also citations to other types of legal document. Paragraph G on page 272 cites the statutory text contained in the Trade Descriptions Act, 1968. Interestingly the citation takes two forms, a relatively informal presentation within the text:

section 14(1)(a)(ii) of the Trade Descriptions Act 1968

and the more standardised form in a footnote to the long headnote of the case:

Trade Descriptions Act 1968 s. 14(1).

The texts cited above may, in turn, cite other texts in a recursive manner and the searcher can follow conceptual paths through the law before returning in his dialogue to an earlier position. This navigation can be considered as a form of hypertext and is greatly assisted by symbolic identification of units of the text.

With symbolic keys, each unit of text is identified uniquely by its position in the framework of legal sources which enables the searcher either to display directly text by entering its symbolic identifier or to refer to it from another text by asking the system to resolve a reference displayed on the screen. With a physical method for identifying units of text such as buttons or pointers, the binding between one text and another is early, the order of loading of texts is critical and permissable data volumes are low (10-20Mb) because of the difficulties of authorship. With the logical method of symbolic keys, binding is late, texts may be loaded in any order, there is no limit on data volumes and referential transparency can be readily achieved. The difference should be noted between symbolic keys and the labels provided by many systems to names units of text: the absence of primary rules controlling label values prevents the construction of integrated cross-referenced texts.

An outstanding example of the demand for referential transparency is found in dictionaries and encylopaediae. These works of reference do not themselves usually provide the full text of the primary source but may often refer to full text documents to be found elsewhere. If a question arises on the meaning of a legal term as part of a legal problem that a lawyer is engaged on for a client, the lawyer might turn to Osborne's specialist dictionary [Heather & Rossiter 1988] for law as a starting point but a single dictionary entry would rarely provide a final answer. Cross-references to other entries in Osborne require the reader to turn to these. An electronic dictionary should be able to have these ready for display or even have them all concurrently displayed. However, Osborne also refers to external references such as statutes and cases. In an all-encompassing integrated full text medium such as the 'docuverse' envisaged by Nelson [Nelson 1988], these references should also be automatically retrieved without the user having to initiate fresh searches.

Other dictionaries present similar opportunities and problems. Thus the Oxford English Dictionary contains references both to other dictionary entries and to other sources in which examples of usage are cited. Raymond and Tompa show the problems with addressing a variety of external sources in a hypertext environment [Raymond & Tompa 1988], many of which would be at least partially solved by the use of symbolic keys in a database approach.

3.2 Human Factors

The requirement for symbolic keys extends the searching requirements mentioned earlier. The symbolic key comprises a polynomial whose simplest representation on the machine is a series of formatted fields holding integers. Any representation is possible but most database implementations operate with a direct mapping from primary keys to some internal quasi-physical address within the system. Manual systems use alphanumeric keys. For instance a section in UK statutes is usually referred to by the name of the act, a year and the number of the section as in the earlier reference to section 14 (subsection 1) of the Trade

Descriptions Act of 1968:

Trade Descriptions Act 1968 s. 14(1).

However, the years relate to sessions of parliament which cross calendar years and before 1963, these were also expressed in terms of alphanumeric regnal years as for example:

Housing (Financial Provisions) Act 1958, 6 & 7 Elizabeth II.

Earlier acts make citations only using regnal years. Thus the Disabilities Repeal Act (c. 43) 15 & 16 Victoria has the following preamble stating the intention to repeal earlier legislation in the regnal years 1 George I (1714-5) and 6 George III (1765-6):

"An act to repeal certain disabilities under the First of George the First, Chapter Thirteen and the Sixth of George the Third, Chapter Fifty-three."

This situation illustrates the transitional problems in moving from old forms of representation to new. Information already in existence cannot arbitrarily be changed retroactively to bring it into line with some present-day standard. Even where in principle it is possible to convert to some universally acceptable form such as calendar dates, there are many situations where there is no natural transformation to a numeric form as with legal cases which are referred to by the names of the parties. For practical efficiency, the terms in a polynomial symbolic key need to be alphanumeric and not just integers. The Queen's printer, HMSO, in parallel with their printed version also prepare data tapes and to correspond with the manual identifier 1925 c.26 employ a form including an unprintable character:

$$1925c. < ESC - D > 26$$

It is possible to use this form as a symbolic polynomial [Heather & Rossiter 1987a] but it has been found that while the manual format was appropriate for human processing without machines, it was not necessarily appropriate for human processing with the machine. The polynomial symbolic key used in our work is shown in Figure 2. This gives the features necessary to identify uniquely each unit of text and to order the units as in the printed version. Further, as will be discussed later, addressing component values of the key provides the basis for data manipulation operators such as aggregation.

3.2. 1 Cognitive Textual Types

The underlying objective in the use of text is ease of human expression and communication. But for a machine, ease of expression and communication involves quite different features from words to be found in text, rather words

Figure 2. Polynomial Symbolic Key for Legal Text

```
---------------------------------------------------
| SI | year | chapter | header | part no.| part no.|
|    |      |   no.   |        |  start  |  high   | ........
---------------------------------------------------

--------------------------------------------------
| schedule | schedule | subschedule | subschedule |
| no.start | no.high  | no.start    | no.high     |  .......
--------------------------------------------------

-----------------------------------------------------------
| section or para- | section or para- | subsection or sub- |
| graph no.start   | graph no.high    | paragraph no.      |  ...
-----------------------------------------------------------

--------------------
| duplication No.  |
--------------------
```

---

which are sets of atomic data that can be manipulated by logical operations. For humans, text expresses natural language and conveys meaning and understanding. Much of the cognitive value of text that enables the human mind to process its semantic and pragmatic contents is provided in complex structures and relationships between words.

Natural language syntax, the third structural domain recognized in basic linguistic theory, is treated intuitively and usually unconsciously by humans but is always implicitly present even in the most simple of expressions. Syntax is much more amenable to computational methods and it is sensible to exploit this to the full by making explicit in every way possible all syntactical relationships in text. Mathematical models, like predicate logic [Sparck-Jones & Wilks 1985] can assist with parsing but may give rise to problems of normalization and loss of information in automated processing. Very important are the computational tools of data modelling and abstract data typing. Indeed it seems that computer models to analyse text structures are a preliminary requisite for the use of the mathematical methods like predicate logic. For instance it is important to anticipate the types of text in cognitive human terms.

The text in Figure 1 is classified into the four different categories of whole-text, short-text, loose-text and force-text. Whole-text is unmarked, short-text is enclosed in angular brackets, loose-text is underlined with a broken line and force-text is underlined with a solid line. Whole-text is the ordinary full and free expression of language and conforms to the accepted rules of such communication. Much of the text in Figure 1 is of this form as from paragraph D on page 272 to paragraph C on page 273. Short-text is an abbreviated form of whole-text in which the strict rules of grammar are relaxed: short-text usually relies on knowledge common to the reader and writer for interpretation. The terse information in paragraph H at the foot of page 273 and in paragraph G on page 274 is of this type being incomplete grammatically but readily comprehensible to a lawyer. Loose-text consists of fragments of text in note form such as headings, an example of which is the heading in paragraph A on page 272.

Force-text is text in which the writer is tightly controlled as to what may be expressed. The need to distinguish between vocabulary that is either 'free' or 'controlled' was recognized in an earlier report [Heather & Rossiter 1987b]. A considerable number of the text entries in Figure 1 can be considered to be of the type force-text:

- attributions of respondent or appellant (page 272, paragraph A),
- date of publication of case i.e. 1985 (272-A),
- where the case was heard i.e. House of Lords (272-A),
- dates on which the case was heard (272-B),
- the Lords of Appeal hearing the case (272-B),
- letters identifying sections within the case report A-H (272-A),
- the keywords and phrases assigned to the case (272-C),
- the footnote identifying the cited act (272-H),
- the title of the case (273-A),
- standard words of attribution to specific Lords (for example 273-G: "Per Lord Templeman"),
- forward-pointers to subsequent parts of the case (for example 273-F: "post, p.298F-G"),
- lists of other cases cited (274A-F).

An interesting hybrid form of text occurs where the whole- or short-text contains passages which can also be viewed as force-text. Thus complete references to statutes (272-G) and cases (273-H) are made in the main text. The content of these references is logically constrained but the format is less formal than usual. Some other references e.g. "section 24 of the Act" in 275-H are readily comprehensible to the human mind which can easily make the necessary currency adjustments but are much more difficult for the machine to assimilate automatically. A passage within the whole- or short-text which cites other text has only been considered to be force-text if it can be unambiguously resolved by the machine without the assumption of currency indicators.

The types of text described above can be regarded as true data types for they represent different forms of text, each with its own set of operators for semantic comprehension. Clearly the ease with which the machine could capture the semantics of each type varies considerably. The meaning of force-text in its use in symbolic identifiers, dates, and keywords can be deduced by programs in one or two passes but that of whole-text requires extensive natural language processing.

3.2. 2 Text in Human Behaviour

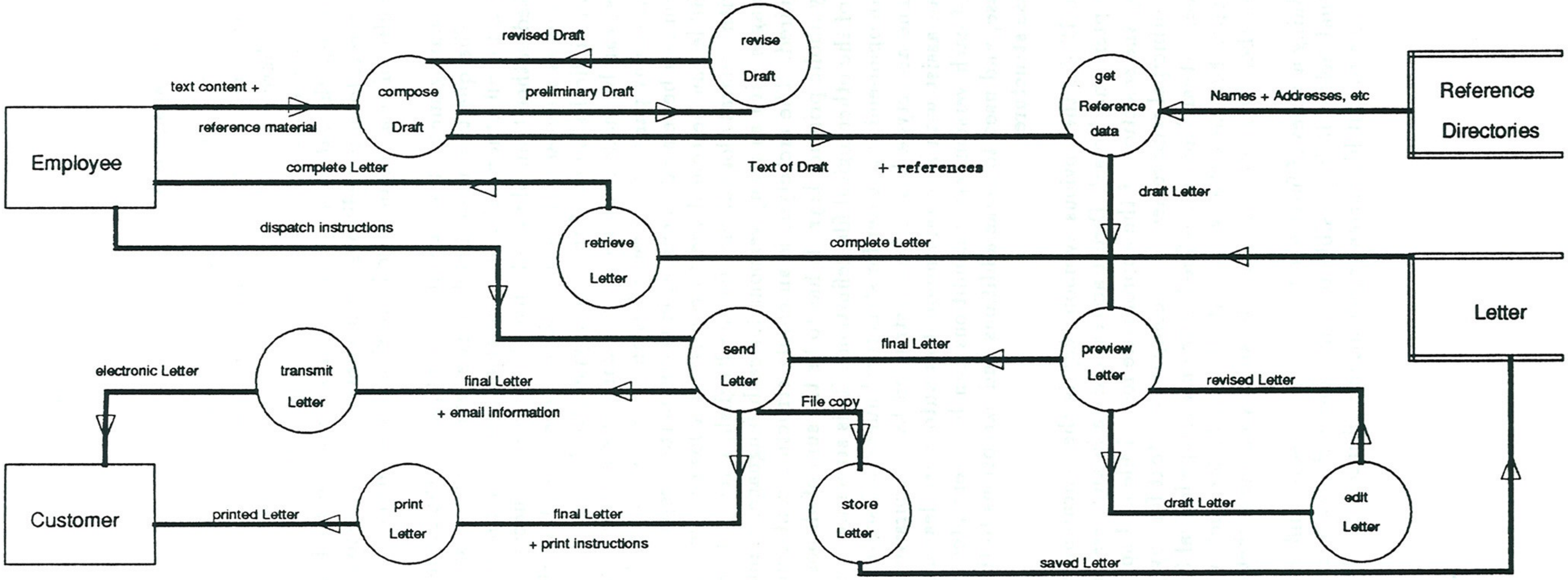
There are other demands made by text applications beyond the static forms so far discussed. Whilst some bodies of text such as law are relatively static, others such as News services involve very high volume of transient data with constant additions of the most recent data, archiving of older data and in-place modification of existing data. Although such updating poses many problems for efficiency in use of disk space and CPU time, it presents few conceptual difficulties. However, in other areas such as the electronic office, there are dynamic structures to be considered which are of much greater significance for database technology. Thus the flow of a letter through an office system can be represented by the data flow diagram shown in Figure 3 [Diamandis 1986]. Data flow diagrams are made up of four basic elements:

- Data flows, represented by directed lines. The arrow represents the direction of the data flow.
- Processes represented by circles.
- Storage of data irrespective of the kind of files in which it is stored. Thus, stores of data could be either database files or ordinary files. They are represented by open rectangles.
- Data sources and sinks, external entities acting as terminators of the global process, represented by rectangles.

Thus, besides the simple in-place modification techniques traditionally used in data processing, there is clearly also a need to control the behaviour of an object through its life-cycle in the drafting process. Transaction processing in the standard DBMS environment, in which units of work are defined that either totally succeed or totally fail, can provide control over the life-cycle of an object. However, even in relatively simple areas such as banking applications, there are locking problems in complex transactions when concurrent access by a number of users to records being modified is temporarily prevented. In the long life-cycle of a report, locking problems would become severe with no ad hoc access possible once the transaction had commenced and the complete back-out of all work done if an error occurred at any point. Breaking the transaction up into a series of smaller transactions would result in control over the whole process being exercised at a much lower-level, for instance, algorithmically using a program

Figure 3.

Data Flow Diagram showing Behaviour of the Entity 'Letter'



written in a host language such as Pascal or C. A more modular method of control in a robust declarative language is ideally required.

3.2. 3 Version Management and Views

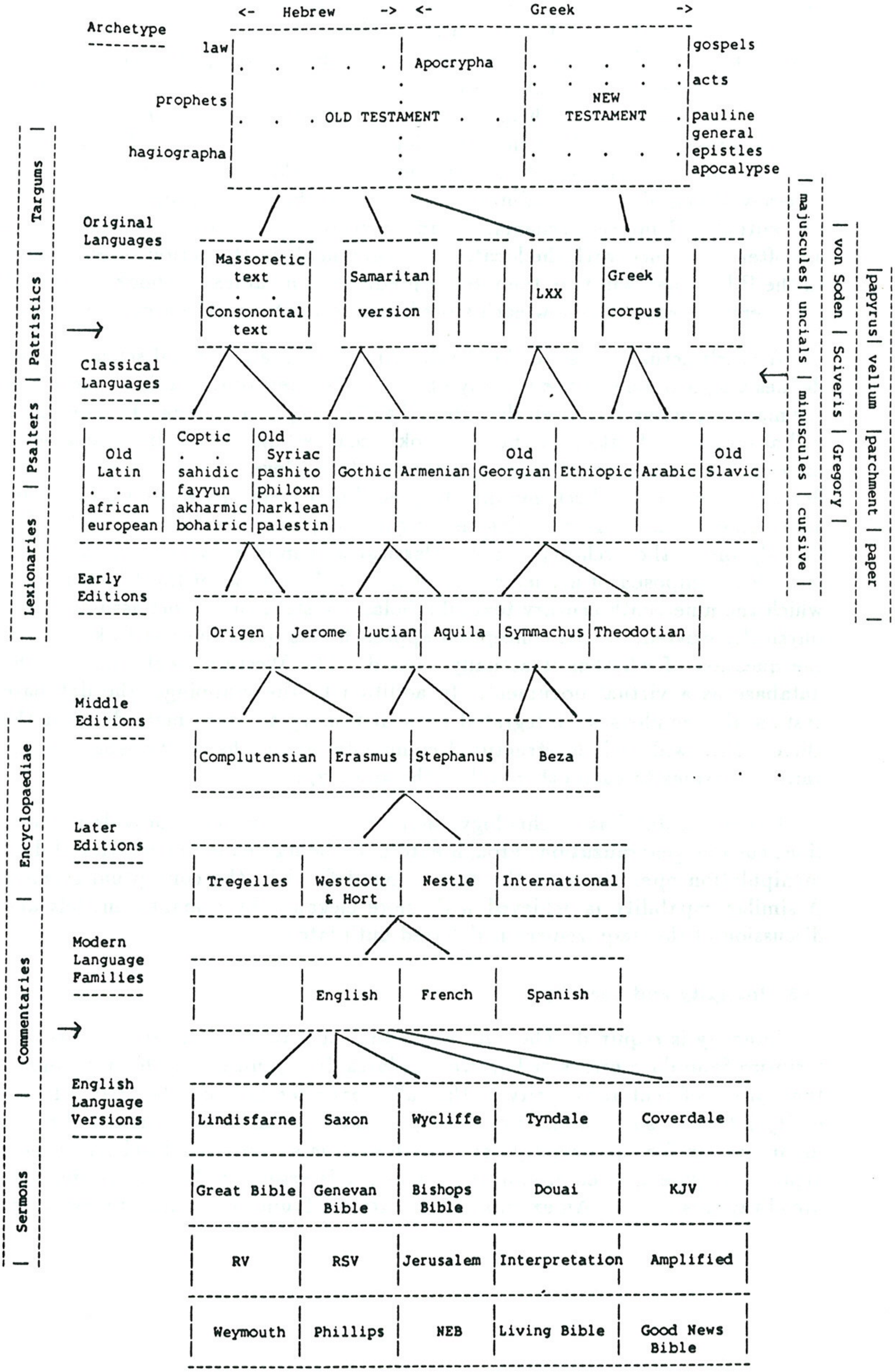
With any text document but more particularly with ones which may take many months or even years to complete, there is also the problem of version management where different versions of the same text can be extant at a given time. Clearly the status of a report with respect to modifications should be available through the database system but this is not the whole problem : users will need to revert to earlier editions of a document from time to time so temporal aspects are also of vital importance in version management.

Version management also plays an integral part in the manipulation of parallel texts. The question immediately arises which version should be used as the reference for all others. Because the electronic medium is essentially logically oriented rather than physically based, there is no necessity to select any actual version as standard. It is possible to develop further the concept first pioneered in biblical studies of the 'archetype' [Heather & Rossiter 1989b]. In this way, use can be made of an abstract document structure independent of the text content in any particular version but an archetype to which the text of any given version can be related. Thus, as in the hypertext representation of Figure 4(a), the Bible can be considered to consist of a logical file for the archetype together with separate logical files for the English, Greek, Hebrew, Latin, and other languages with as many versions for each language, manuscript families, etc, as required. The archetype is an empty structure containing no text data and only symbolic primary keys. It has to be the sum of all possible versions as a superset of the other logical file definitions. This structure is never closed and must remain open-ended to cater for structural nuances discovered later in other versions. The archetype has to be constructed by a recursive process and the machine can assist with a certain amount of automated learning. Thus during the build of each version, error conditions can be intercepted and collected in the database, and used to make additions and exceptions to the archetype and the database structure.

For the Bible, the obvious symbolic key is the concatenation of the fields 'book', 'chapter' and 'verse'. Direct access to data in any version can be readily obtained by the selection of the appropriate logical file and specification of the full key. A principal requirement is to compare the text for any given verse. For this purpose, virtual fields are defined in the archetype logical file which convert the archetypal key into the 'verse.text' data held for each version. Parallel versions can then be readily displayed by specifying the archetypal key.

By defining a series of further subfiles, each accessing different combinations of the virtual fields, it is possible to tailor quite closely a user's view of the biblical sources to suit his current application and interests. Thus another way

Figure 4(a). Hypertext Representation of the Bible.



of looking at the implementation is that it provides different external views of an archetype which can be thought of as the single conceptual schema for the Bible. Such a model is well represented in Figure 4(b) in terms of the standard ANSI/SPARC architecture [Tsichritzis & Klug 1978] developed to provide views of data on three different planes in the database environment. At the lowest level, there is an internal schema to describe the storage structure and contents of each version of the Bible held in the database. The conceptual schema defines the structure and contents of the reference form of the Bible, i.e. the archetype. A series of mappings are maintained to indicate the relationship between each conceptual and internal schema. In administrative databases, these mappings are often concerned with the location of stored fields in the various files on disk. In the Bible, they will take the form of predicates on values for 'book', 'chapter' and 'verse' to represent how each stored version relates to the archetype.

A user's actual access to the database is through an external schema which defines a logical view of the underlying data. The view obtained is a function of the mapping defined between the external and conceptual schema. This mapping will also be a predicate on values of 'book', 'chapter' and 'verse' in an analagous manner to that between the conceptual and internal levels. To achieve maximum flexibility, the external schema may be placed in a hierarchy as shown in Figure 4(b). Thus a scholar in studying the original may view the text mapped either directly on to the archetype or via classical and modern views. An example of a view composed of a selected set of text is the notion of the Western Text which the nineteenth century textual scholars Westcott and Hort assumed once physically existed. In the conceptual approach it is quite natural to keep open the question of whether it actually existed. The Western Text exists in the database as a virtual document. In addition to the mappings, the database system also employs an integrated data dictionary to store meta-data on the Bible. This will include structured commentaries on the relationships of the various versions to each other and to the archetype.

In current database technology, another use of views is to provide abstractions such as generalization through external schema defined as a series of data manipulation operations on the data items defined in the conceptual schema. A similar capability is achieved with more elegance by semantic models and discussion of this requirement is deferred until later.

3.3 Integrity and Security

Integrity is required. The data needs the usual protection through recovery facilities from data errors or hardware failures during updates. Of equal importance are: referential integrity so that all cross-references can be resolved; and entity integrity so that the uniqueness of entities can be guaranteed through unique values for the primary key. The former is an important aid for authors creating a hypertext environment in which it is essential that cross-references are always resolvable. An example of the latter is found in the statute law where

Figure 4(b). Schema of Bible Versions (ANSI/SPARC Architecture).

```
             Original-Classical view
             ----------------
             |defn. Original|
             | record-type  |
             |--------------|
             |Mapping Orign.|
             | -> Classical |
             ----------------
                   ↑
                   |
                   V
             Classical-Modern view
             ----------------
             |defn. Class.  |
             | record-type  |
             |--------------|
             |mapping Class.|
             | -> Modern    |
             ----------------
                    ↑
multi-              |
level               V
external  Modern-Arch.view    Classical-Arch.view   Original-Arch.view
schema    ----------------    ----------------      ----------------
          |defn. Modern  |    |defn. Class.  |      |defn. Orign.  |
          | record-type  |    | record-type  |      | record-type  |
          |--------------|    |--------------|      |--------------|
          |mapping Modern|    |mapping Class.|      |mapping Orign.|
          | -> Archetype |    | -> Archetype |      | -> Archetype |
          ----------------    ----------------      ----------------
               ↑                    ↑                     ↑          ↑
               |                    |                     |          |
        mappings to conceptual Archetype and between external levels |
               in form of predicate on Book, Chapter, Verse   ↑      |
               |                    |                     |   |      |
               V                    V                     V   |      |
conceptual  ---------------------------------------------
  schema    |definition Archetype conceptual record-type|   integrated
            |-------------------------------------------|<--   data
            |       mapping Archetype -> internal       |   dictionary
            ---------------------------------------------
               ↑                  ↑                 ↑            |
               |                  |                 |            V
        mappings to stored record-types in form of predicate on Book,
                                Chapter, Verse
               |                  |                 |
               ↓                  ↓                 ↓
internal  Hebrew Massoretic    Greek LXX          English KJV      ... etc.
 schema   ----------------  ----------------  ----------------  (other
          |defn. stored  |  |defn. stored  |  |defn. stored    |versions)
          | record-type  |  | record-type  |  | record-type    |
          ----------------  ----------------  ----------------
```

apparent duplicate records occasionally occur with the same symbolic key because certain nuances in the layout of the text are not fully captured in the data structure. The effort required to modify the data structure may not be justified to enable perhaps another 0.1% of the records to have unique symbolic keys. However, it is essential whilst adding data to ensure that the apparent duplicates are detected, added to the textbase and flagged as non-conforming data.

Value integrity should also be enforced. In its simplest form, this can be achieved by defining a domain of acceptable values that a data item may take. Thus in the textbase holding information on English criminal offences and their penalties [Rossiter et al 1988], codes used to represent categories of criminal intent and presumed public policy should be held in a domain holding all valid values. Besides the use of domains, value integrity can be achieved through many other means such as abstract data typing, stop and go lists, and thesauri. More complex integrity constraints are required in practice in some situations. For example, in the statute law data tapes provided by HMSO, the text formatting attributes indicated in the markup language are not assigned generally but are applied only in certain contexts and these need to be enforced in construction of the textbase. For instance, the declaration of footnotes for a statute is performed within each act shortly after the global headings and any later declarations should be flagged as potential errors.

Another feature required in an operational environment is security. It should be possible to restrict access by users to whole files or on a more selective basis to designated fields or indexes, with the ideal situation being to utilize data values as an additional data-driven means of granting or denying access.

3.4 Textual Applications

Besides retrieval and update, users may examine texts to produce word frequency lists and distributions of frequencies, and to apply statistical tests [Davis 1983] to analyse word co-occurrences, collocations [Linton 1982] and sentence length. Thus, texts of Shakespeare and of the Bible have been analysed in great detail. However, surprisingly, most of this analysis has not been performed using database systems in spite of much of the information such as word frequency counts being capable of being held in database tables. The main tool has been packages such as the Oxford Concordance Program which construct the necessary tables from scratch in each analysis of a text.

A clear difficulty in using database systems for analysis is that each application involves subtleties such as a special way of combining textual fields or of handling word stemming which the database kernel is unable to satisfy directly. To handle such subtleties, the user must embed calls to the database system in a host program written in a high-level language such as Pascal with abstract data typing or in SNOBOL with text editing facilities. This separation of data from function is an inherent feature of current data base architecture [Bloom & Zdonik 1987]. There are clear advantages to be gained by integrating more closely the host languages and the database kernel so that, for instance, the database system would automatically adopt the data typing of a Pascal program or provide string manipulation functions for text transformation during update, display, indexing and the construction of virtual textual data items.

3.5 Natural Data Loading

If textbases are to be quickly designed and built by users who have few traditional skills in programming, data loading should use techniques that do not involve substantial amounts of editing or procedural programming, The loading of textbases follows hierarchical paths in the structure of the text and implicit assumptions based on currency need to be made about the values for the constituents of the symbolic key of each unit. Input languages which modelled this structure in a natural manner would assist implementations. The use of 3GL host languages is common in this task but there is an impedance mismatch between the database and programming language data structures. The variables used in the programming language are not directly related to data field values in the database and integrity is therefore enforced indirectly. To achieve a natural capture of the hierarchical structure, the ideal is for automatic value inheritance whereby the identifier of an entity is in part automatically inherited from others which appear at higher levels in the tree structure.

3.6 Data Structures and Models

Textual data structures are described later in terms of semantic models and class structures of objects. Examination of applications at Newcastle shows that text plays an important part in many research databases [Rossiter et al 1988]. The natural history data of the North of England held by the Hancock Museum employs text at a vital elementary level: a text field is used to supplement highly formatted data to record comments and observations which do not fit the highly-structured classification scheme designed at the outset. Similar considerations apply to the data held on Chinese provincial politicians by the Department of Politics where formatted data on tours of duty is supplemented by less structured biographical data. Medical research databases, in addition to storing many items of numerical and coded information on subjects such as biochemistry, can hold large bodies of text to record complete details of operations, surgical procedures and unexpected complications. Genealogical databases involve complex relationships between people with identifiable objects such as branch, family, marriages and individuals. Uncertainty over exact relationships means that textual evidence such as wills and litigation should be recorded verbatim.

In a further broad class of applications, text comprises the most important part of each record. Thus in Hansard, the main data is the verbatim record of business in the UK Parliament although ancillary classifiable data such as dates and names of bills and speakers also need to be held in formatted form. Identifiable objects in Hansard include days of business, debates, speakers and speeches [Hudson 1985]. In the Bible, the text held in the verses is the critical part of the data. Identifiable objects are testament, book, chapter and verse, and a symbolic key comprising book, chapter and verse is a natural obvious method of addressing verse data. However, for some biblical applications such as a commentary, where opinion and factors influencing judgement are considered, the structure of book, chapter and verse is too coarse: words and their variants or even individual characters are the basic units [Heather & Rossiter 1989b].

The applications so far discussed in this section basically involve text in hierarchical structures with a choice of unit size for manipulation ranging from word or character up through various intermediate text units to documents in their own right such as the New Testament. This gives rise to the major problem of the choice of unit size.

3.6. 1 Unit Size

Unlike informal systems which might store text as a continuous character stream, formal systems need an object unit [Heather 1986]. Traditionally a choice has had to be made which is usually governed by the storage capability of the system (e.g. track and block sizes, buffer restrictions, spanning facilities, etc), by the human capacity for searching, retrieving and comprehending the information [Blair & Maron 1985] and by the character of the document which though will have often been 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, in principle, to retain the ability of natural language to keep the choice of unit dynamic and with the option of lazy evaluation to have the power to postpone any decision until the full circumstances and context of the use are known.

3.6. 2 Non-hierarchical Text Structures

There are texts for which hierarchical structures are inadequate. Shakespeare and legal texts are good examples. Such texts are later represented in terms of semantic models. 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.

3.6. 3 Generalization and Specialization

Linked particularly with the navigation requirements described earlier is the need to generalise when describing text structures. Thus, for example, in a hierarchical text structure, any one part of the tree may usually cite any other part. The textbase will need to be viewed at two levels: generalisation for an overview in which any type of text object cites any other type; and specialization for the system implementation in which a specific type of text object cites another specific type. The two levels provide abstract and concrete views respectively of the citation process and are discussed in more detail later when semantic models are introduced.

3.7 Multi-Media Systems

Of increasing importance in the future will be the storage and manipulation of textual, image, formatted and other forms of data in an integrated fashion by a single system. For example, in computer-aided manufacturing (CAM), the image data is often accompanied by text for specifying semantic detail which cannot be readily expressed in a graphical manner. In examining new approaches to textbases, it is therefore essential to bear in mind that an integrated holistic approach is needed to handle the different sorts of data. Image data in systems such as CAD/CAM are discussed in the next section to illustrate the similarities and differences between text and image data. However, even in pure text systems some of the principles of a multi-media approach may need to be adopted. Thus in the study of politics, diplomacy and literature, multilingual textbases involving languages such as Russian, Chinese, Greek and Japanese present some of the problems of multi-media systems with the need to hold both ideographic representations of their character sets and to handle diverse interpretations of such basic text entities as characters and words.

3.7. 1 Structures in Engineering Data

The problems in manipulating engineering data by relational systems [Lorie 1981] have been established for some time:

1. In drawings, graphs, and designs, the prevalent structures are represented in the form of complex objects.
2. If relational database systems are used for storing and manipulating the data, there are severe performance problems in reconstructing designs on screen from data held in relations. These result from the large number of joins required to reconstruct data which have been excessively fragmented through the normalization process. The poor performance is of critical importance in CAD/CAM applications because of the long time taken to draw designs on screens.
3. So far less emphasis has been placed on searching for particular features than in text systems but there is likely to be an increasing need to match on certain shapes. Work by Eakins [Eakins 1989] has demonstrated the potential and

problems of shape matching. To facilitate retrieval, surrogate representations are made of the images and matched against similar constructions of the user requirements. Whilst this technique enables searches to be made and items retrieved, there are problems which are analagous to those in information retrieval of bibliographic items: perfect recall and precision is not possible because matching is being performed at a syntactic rather than a semantic level.

Besides identifying the problems with conventional database technology, Lorie [Lorie et al 1985] attempted to adapt the relational model to manipulate engineering data. To augment the native atomic structuring of the relational model designed for simple administrative objects, Lorie proposed extensions to provide molecular structures more appropriate for representing the complex objects described above. The molecular structures serve two purposes:

- to structure objects hierarchically; and
- to improve performance through the provision with each object of reference attributes containing backward and forward pointers between each level in the tree.

The molecular structures are not, however, a complete solution to all problems in the manipulation of complex objects. The functionality of the reference system is strictly hierarchical so that shared subobjects are not handled well and, even with the pointers, poor response times can occur if considerable aggregation is required.

3.8 Summary of Requirements

Figure 5 summarizes the requirements for textbases from the viewpoint of users. At present, there is a piecemeal approach to satisfying these needs with, for instance:

1. free text retrieval systems concentrating on Boolean searches and very large record sizes,
2. database systems emphasising safe updating with the rigorous maintenance of integrity and security and providing alternative views of the same stored data,
3. hypertext systems specializing in methods for navigation, and
4. expert systems attempting to interpret the semantics of the text in terms of rules.

The effectiveness of these approaches in satisfying user requirements can be evaluated by considering examples of the way they have been applied to legal texts.

4: Approaches to the Modelling and Manipulation of Text

4.1 Free Text Retrieval Systems

Because lawyers deal in words, the law early attracted the attention of workers in the field of text. One of the first attempts to manipulate law by computer involved the design of free text retrieval systems such as STATUS, STAIRS and LEXIS. These systems were closely tailored to contemporary ideas of the requirements of users. Thus it was perceived that textual structures were basically hierarchical of the form for statutes of:

$$chapter -> paragraph -> sentence -> word$$

A basic requirement was fast access on words contained within the text so priority was given to the development of inverted files to hold word indexes and for the operation of Boolean logic. Figure 6 shows the hierarchical record structure of a statute together with the typical structure of an inverted file to provide fast access in word searching. The inverted file structure mirrors the hierarchical structure of the text and can readily provide matching on two or more words in context through appropriate selection of columns for examination. Thus, in what would be termed a projection in relational systems, a search for the two words X and Y in a section would use the columns WORD, CHAP#, YEAR, P# and S# and ignore columns SS# and POSITION. A search for X and Y within a particular number of words of each other would use all columns in the inverted file.

In STATUS and STAIRS, the main ambitions are to achieve dynamic control over the unit of data searched and to provide convenient interfaces to the end-user so that a high level of abstraction is presented [Bain et al 1989]. Unfortunately, there is a lack of unifying principles and an emphasis on improvisation in achieving these objectives. Thus three kinds of file will be found in a typical free text retrieval system: base text, index and thesaurus. Each will be defined and manipulated in a different language so that, for example, the inverted file is not a fully-fledged textbase table available for further applications and a special language is required to access the thesaurus. In ANSI/SPARC terms, the systems employ a single-level (flat) architecture at a level intermediate to the internal and conceptual schemata. This reduces flexibility by not allowing mappings to be introduced between the various ANSI/SPARC levels to provide resilience and extensibility.

General filing techniques such as cross-referencing, symbolic identification of data units and support for a wide range of data-types other than text are available only in rudimentary form. It is thus difficult to capture all structural

Figure 5. Requirements for Textbases from Users' Viewpoint

1. Design of STRUCTURE for holding text
   - unlimited size of fields and records
   - symbolic identification of records
   - data models
     - hierarchical
     - non-hierarchical
   - dynamic control of unit size
   - generalization and specialization
2. Various formats for DISPLAY
   - human
   - machine-machine (wp, mark-up)
3. RETRIEVAL
   - fast
   - non-procedural interactive languages
   - words + phrases in text
     - context
     - proximity matching
   - keywords
     - free vocabulary
     - controlled vocabulary (thesauri, stop, go)
   - 'formatted' data
   - identifiers of text (symbolic key)
4. NAVIGATION through texts following conceptual paths
   - referential transparency (hypertext)
5. SEMANTIC factors
   - parsing
   - predicate logic, machine translation
   - cognitive textual types
6. TEMPORAL management with consistent updating
   - in-place modification, addition of data, archiving
   - dynamic behaviour - control of document life cycle
   - version management
   - concurrent access
   - value inheritance for natural data loading
7. VIEWS
   - parallel texts
8. INTEGRITY
   - protection against hardware failures
   - referential
   - value
9. SECURITY
   - whole file
   - designated fields
   - data driven
10. Textual ANALYSIS
    - function integrated with data
    - word frequency lists, distribution of frequencies
    - statistical tests e.g. sentence length
    - word co-occurrences
11. MULTI-MEDIA
    - integration of text and other data (unified model)

Figure 6. Word Index in a Free Text Retrieval System

```
    STATUTE record-type

         CHAPTER
         |    ----------------------------------------
         |    |   CHAP# |  YEAR  | HEADINGS | TEXT   |
         |    ----------------------------------------
         |
         V
         PART
         |    ------------------------------------
         |    |   P#           |     TEXT        |
         |    ------------------------------------
         |
         V
         SECTION
         |    ------------------------------------
         |    |   S#           |      TEXT       |
         |    ------------------------------------
         |
         V
         SUBSECTION
              ------------------------------------
              |   SS#          |      TEXT       |
              ------------------------------------

  Inverted File

    ------------------------------------------------------
    | WORD   | CHAP# | YEAR | P# | S# | SS# | POSITION  |
    ------------------------------------------------------

              -------------------------------------
```

detail in textbase design or to perform complex data manipulation. Whilst there is flexibility in selection of unit size for searching purposes, there is considerable inflexibility in the choice of unit size for other purposes. In all systems for the purpose of storage, a document size must be selected at the outset which defines the maximum unit for searching. The choice is essentially arbitrary. In statute law, the choice of a document size as an act could involve data transfers of 2-3 Mb so that some smaller unit must be selected with profound future implications for the running of the textbase. It is symptomatic of systems which only just achieve their initial specification that they fail in new subject areas. Thus STATUS designed at Harwell for a small body of atomic energy law was not equal to the task of providing a full service with the whole body of English statutes and the commercial venture EUROLEX that attempted this failed.

LEXIS is a well-known free text retrieval system for law. It is dedicated to this purpose and has not been generalized in the same way as STATUS and STAIRS for handling applications other than full text. The system provides fast matching on words in proximity to each other and in the context unit of a section through the storage of word positions in inverted files in a similar manner to that shown in Figure 6. However, the storage of what amounts to a physical address of every word (except stopwords) leads to heavy overheads and the user is restricted to some extent by the rigid selection for the basic document unit of a section. The user cannot readily manipulate logical structures involving aggregation of data at higher levels: the system has had to be patched to provide this facility for statutes. Furthermore, structural information, such as identification of documents which could have been used for symbolic key purposes, is not explicitly held in a classified form: section, schedule and other numbers are held as free text within segments rather than as formatted values of type integer. It is thus difficult to make cross-references from one text to another. Finally, there is a heavy reliance on manual editing to convert data into a suitable form for inputting to the system. It can thus be concluded that the system is not extensible to handle novel applications and, in general, there is a failure to exploit the benefits of automation, leading to high costs and under-usage.

4.2 Database Technology

A greater emphasis on logical form is found in database systems. Briefly summarized, the main principles of database technology in this context are:

1. to create and maintain reliably databases holding large amounts of data (1-500Gb) in a persistent storage medium.
2. to employ structured semi-formal methods to model the data at various levels from semantic models, which provide highly abstract views, through logical models, such as a relational design, to physical data structures, which facilitate fast access for retrieval or modification.
3. to provide a non-procedural programming environment in which high user productivity can be achieved,
4. to prevent unauthorized access to all or part of the data,
5. to provide concurrent access both for reading and writing, and
6. to maintain the integrity of the data as far as possible at all times.

Traditionally there has been an emphasis on formatted data with the support of only a limited range of data types such as integer, real, character and date.

4.2. 1 Work at Newcastle

In Newcastle, the database management system (DBMS) SPIRES has been employed on a series of text-based applications since 1978. SPIRES can be thought of as a semi-relational DBMS with flexible data-typing and indexing facilities. It is semi-relational because it enables cross-references to be made by value rather than by pointer but does not employ relational set languages. It is a DBMS because it effectively provides a three-level architecture of external, conceptual and internal schemata. The range of data-types includes a series of some 30 pre-defined types (integer, real, word, date, personal name, institutional name, etc) plus the facility for user-defined abstract data types employing the SPIRES PROTOCOLS language which provides most of the standard SNOBOL text editing functions but is also fully integrated with the database kernel. Data-typing is enforced by action rules in the conceptual schema. The abstract data typing and integration of function with data enables SPIRES to be viewed as a rudimentary object base system. Indexing facilities include the ability to construct word indexes with stop and go lists and thesauri. Fast prototyping is available through a File Definer module.

Compared to free text retrieval systems, the data structuring facilities of the SPIRES system enable a greater resolution to be made of the internal structure of statutes. Greater flexibility in data structuring was obtained by defining a non-homogeneous basic unit of record. Sections are used for main text, paragraphs for schedules, footnotes are taken together as a single object and other extraneous information made into objects in their own right. The ability to identify parts of the text by symbolic keys allows users to navigate [Rossiter 1987] from one part of the law to another in a semi-relational manner using the keys as reference points. This is also important for referential integrity in updates. The status of the navigation through the text can itself be held in database records. This provides a complete anaphoric system which hypertext systems are at present attempting to emulate. The retention of all detail contained on the printed page enables new applications to be readily satisfied which had not been anticipated when the database was established. Thus electronic publishing needs can be catered for in a flexible manner, new selective indexes produced and fresh analyses made of subsets of the text. The iterative searching facilities assist users in retrieving parts of the text that are of interest to them using Boolean operators and proximity matching. As described earlier, the three-level architecture can be used to handle parallelism [Heather & Rossiter 1988] and version management with the canonical form being held as a conceptual schema and other versions as external schemata.

4.3 The Hypertext Approach

Because of the importance of document manipulation, hypertext has developed as a separate branch of computing. The same Osborne law dictionary referred to earlier has also been implemented by Wilson [Wilson 1988] in one of the available hypertext systems, GUIDE, which employs directed-graph techniques. It is therefore interesting to compare this with database methods. GUIDE represents logical connections in a physical manner using pointers and buttons. By that method, cross-references may in advance be fully identified as in a network database. In a conceptual database approach to hypertext, links are made dynamically at run-time by the system locating a cited item either within the dictionary database or in external data from some semantic identification of the key value contained in the text. Both means provide for display and navigation through documents. The physically-oriented approach of GUIDE uses less resources but the early binding of identifier to data is more of a static method which allows less flexibility. A more flexible approach is to exploit the dynamic power of late binding using the technology of databases.

4.4 Expert Systems

A very different approach to law and one which has been extensively investigated recently is the development of expert systems which attempt to distil and capture the intent and rules within a particular subject area. Projects [du Feu 1977, Hafner 1981] seeking to represent the knowledge contained in legal text data were current with early expert systems like MYCIN and DENDRAL. One of a long line of recent attempts to formalize law using the logic programming language PROLOG is the expert system for the British Nationality Act where a number of workers [Sergot et al 1986] found it quite some effort to capture only some of the content of one relatively simple Act of Parliament. This example illustrates well some of the problems with this approach. An accurate mapping is not possible between the legal and PROLOG rules, the former being expressed in natural language for the maximum of expressiveness and the latter in clear unambiguous rules for the purpose of logic programming [Leith 1986]. Despite attempts to avoid the problems raised by closed world assumptions in predicate logic, the inherent failure of any formal reductionist system to prove negative facts appears fatal for applications in areas like the law.

Even if it could be achieved, the difficulties of modifying current expert systems to allow for continuing legislation and the effect of court cases on the text are immense. The debate over the likely usefulness of legal expert systems has mainly been between computer scientists [Kowalski & Sergot 1987, Leith 1988]. The only serious use reported in actual legal practice seems to be by practising lawyers like Walter [Walter 1989] who have built their own expert systems which they find very convenient when they themselves are fully conversant with the capabilities and limitations of their systems.

5: Database Models and Textual Structures

The experimental work carried out using SPIRES has demonstrated the potential of database technology for handling text and achieved a favourable environment for formulating principles that can aid the development of a new generation of textbases. Text applications involve large amounts of persistent data of up to 100Gb. The texts need to be structured using a semantic or other model such as the basic relational, network and hierarchical methods. There has to be efficient access both on symbolic keys and on words. The semantic models are more expressive than the basic models and their use will be considered first.

5.1 Semantic Models

There is a range of semantic models that has been proposed in order to incorporate more features and constraints [Smith & Smith 1977] than can be included in the basic ones in an attempt to represent more closely the real world. These include RM/T [Codd 1979], the Chen Entity-Relationship Model [Chen 1976], the Borkin Semantic Model [Borkin 1979], and Taxis [Mylopoulos, Bernstein & Wong 1980]. Text because of its complex nature usually requires a full semantic model and a range of abstract data types to capture completely its structure. Examples of Chen, Borkin and Mylopoulos will be considered here starting with an examination of static aspects.

5.1. 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 7 which follows the style enhanced by Howe [Howe 1983] using rectangles to denote entity-types and diamond-shapes to denote relationships. A relationship flagged '*' is mandatory, otherwise it is optional. In addition the idea of generalisation advanced by Sakai [Sakai 1983] is employed with generic entity-types being denoted by ovals. The generic structure defined here is 'text' which is used to represent the specific entity-types 'section', 'subsection', 'paragraph' and 'subparagraph' as one entity-type if required. Its creation enables for instance the involuted 'cross.reference' relationship to be simplified into an occurrence of the 'text' entity-type may cite another occurrence of the 'text' entity-type. In the absence of the generalization, sixteen 'cross.reference' relationships would be required to handle all the possible combinations: one of the occurrences of the entity-types 'section', 'subsection', 'paragraph' and 'subparagraph' may cite any other occurrence of the entity-types 'section', 'subsection', 'paragraph' and 'subparagraph'.

The smallest object represented in the E-R diagram is 'word' which is a component of the five base entity-types 'section', 'subsection', 'paragraph', 'subparagraph' and 'footnote' but can be viewed more elegantly as a component of the generic entity-type 'text' and of the base entity-type 'footnote' as shown in Figure 7(a). The relationship between 'word' and 'text' is of functionality many-to-many. Each word may appear in many texts and each text may contain many words.

5.1. 2 Class Structures

The structure of the statutes shown in Figure 7(a) can be viewed as the

Figure 7. The Chen Entity-Relationship Model of Statutes:

(a) Diagram

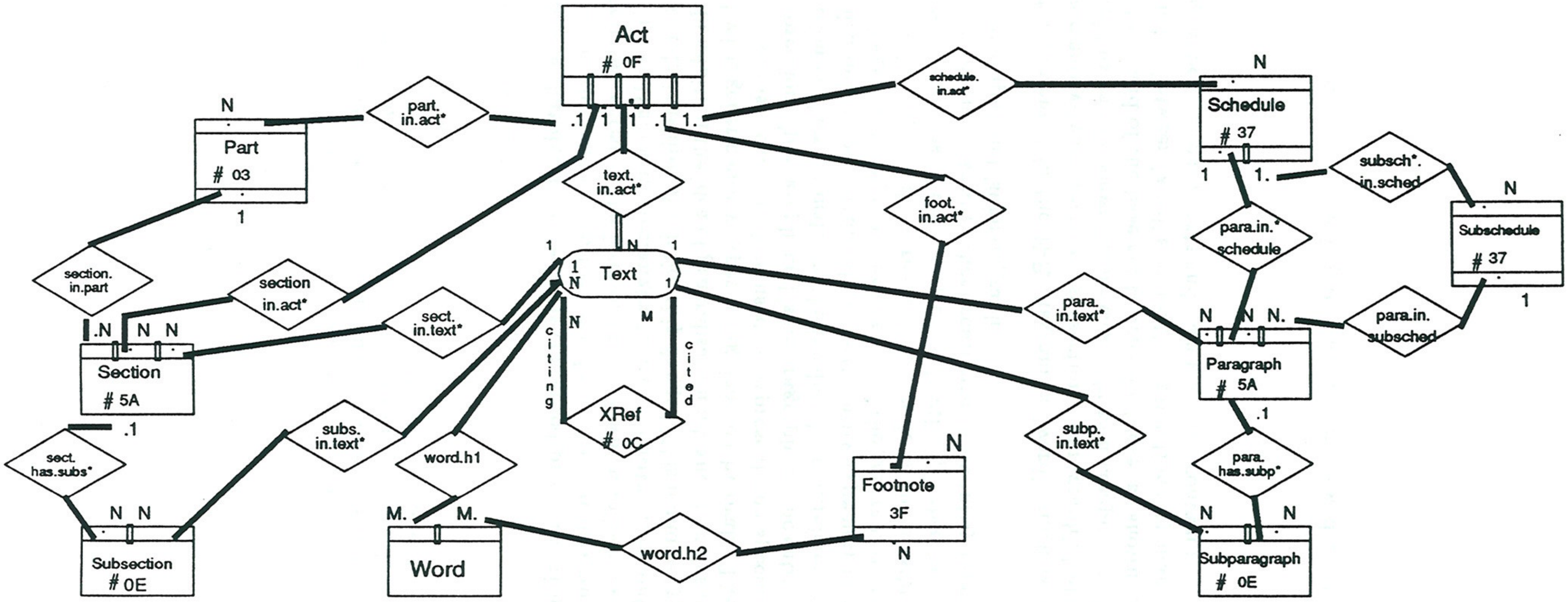


Figure 7. The Chen Entity-Relationship Model of Statutes:
(b) Partially-normalized Table-types

Act (*year, chapter*, title, date, preamble, arrangements,
crossnotes, + 14 text.formatting attributes)

Part (*part#, year, chapter*, part.headings, part.subheadings,
crossnotes, footnotes.to.old.statutes, + 5
text.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.to.old.statutes, + 29
text.formatting attributes)

Subschedule (*subschedule#, schedule#, year, chapter*,
subschedule.headings, crossnotes, omissions,
footnotes.to.old.statutes, + 29 text.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.H1 (*word, position, text.id*)

Word.H2 (*word, position, footnote#, year, chapter*)

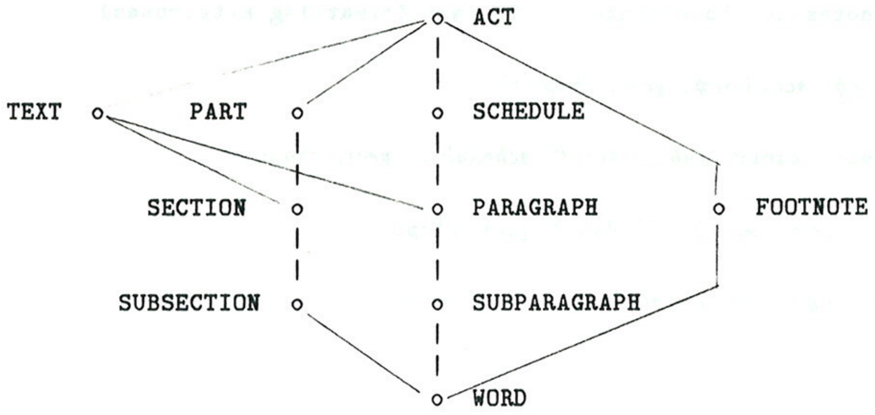
Text {section, subsection, paragraph, subparagraph}

Text (*text.id*, footnote#, marginal.note.other, crossnotes, omissions,
footnotes.to.old.statutes, + 20 text.formatting attributes)

Cross.reference (*citing.text.id, cited.text.id*)
Notes: 1) The italicised attributes comprise the identifier.
2) The brackets define the scope of a generic entity-type.

complex object shown in Figure 8 in which 'word' is a subobject shared by several classes, each word being considered in its own right as well as in its use in a local and global context. The complex object structure illustrates the problem of unit size in which a user may seek to aggregate data at any level of the structure or search for data in the context of any simple or aggregated unit. This problem is discussed in more detail later. Similar difficulties are encountered in other texts such as Shakespeare's plays where the terminology of overlapping fields is used in the humanities to describe the structure shown in Figure 9. The fields are neither contiguous to each other nor contained completely within 2ne another: attributions, type of text and speeches overlap each other with no clear structure other than that they each contain several lines of text. This structure can be represented by the complex object shown in Figure 10. The complex object comprises one subobject 'line' shared by as many as five classes: 'verse', 'prose', 'song', 'stage directions' and 'attribution'. The generalisation 'speech' enables text contained in a mixture of verse, prose and song to be viewed as a single 'speech' entity-type. Work in progress at Newcastle on implementing Shakespeare's plays in a relational database system [Mitchell 1989] shows that the actual data structures are more complicated than those of Figure 10 as a line may contain more than one type of text.

Figure 8. Class Structure for Objects Occurring in Legal Text



The complete E-R model requires more detail than the diagram of Figure 7(a), in particular information on the attributes of each entity-type and which of these comprise the identifier. Figure 7(b) shows this information for the statute law but it should be noted that many of the tables are not fully normalised. Thus many of the non-identifying attributes such as the text formatting ones can have multiple values for each identifier value. To remove such dependencies through further normalisation would require the creation of more entity-types which would complicate the model considerably and produce an un-natural structure.

In other areas, the need for hierarchical tabular structures has been strongly pressed [Dadam et al 1987].

The E-R diagram distinguishes between weak and strong relationships by the positioning of the dot in the upper segment of the entity. Thus 'section.in.act' is a strong relationship in which 'section' must participate if it occurs. When this happens, the section entity automatically inherits the identifier of the act through property inheritance. In a weak relationship such as 'section.in.part', the participation of 'section' is optional and a section does not automatically inherit the part's identifier. This is basically because null values would be posted into a section's identifier, so violating entity integrity when the section was not contained within a part.

The choice of identifiers indicates the numbering system for the statutes, thus section numbers continue to increment across parts whilst paragraph numbers are reset to unity in each schedule but continue to increment across subschedules. For the generalization 'text', the identifier is 'text.id' which can be thought of as a symbolic key similar in form to that defined earlier in Figure 2. The key attributes of the relationship 'word.h1' comprise the word itself 'word' and its logical position in the text as a concatenation of 'text.id' the symbolic key of the addressed text entity and 'position' the physical position of the word within the entity.

The E-R table types in Figure 7(b) can be mapped into a relational system for implementation purposes. However, as will be described later, the transition from the E-R to the relational model involves a loss of semantic expressiveness in aspects such as representing the generalizations.

Other semantic models have been used for modelling text at Newcastle including the Borkin Semantic Model referred to above which is an enhanced relational model. One advantage of this model is that it provides facilities for expressing semantic information by typing the class of a given attribute using what Borkin refers to as a 'characteristic' for each attribute. Thus the rich textual data-typing mentioned earlier can be declared in the conceptual schema as shown in the extract of a complete Borkin model [Heather & Rossiter 1987b] for statute law in Figure 11. The predicate pairs indicate associations and the case is significant: weak links are in lower case and strong links in upper case.

5.1. 3 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. Thus some of the required features for control of the legal drafting process can be expressed in a Taxis-like language by the definition shown in Figure 12. This

Figure 9. Schematic Outline of Text in Shakespeare's Plays

```
Attribution      Type of       Speech            Line
                  Text
 --              --             --
| A1           | Verse        | Speech 1         line 1
|              |              |                       2
|              |              --
|              |              |        2              3
|              |              |                       4
|              |--            |                       5
|              | Prose        |                       6
|              |              |                       7
|              |              |                       8
|              |              |                       9
|              |              --
|              |              |        3             10
|--            |              |                      11
| A2           |              |                      12
|              |--            --
|              | Stage                               13
|              |--            --                     14
|              | Verse        |        4             15
|              |              |
|--            |              |                      16
| A1           |              |                      17
 --              --             --
```

Figure 10. Class Structure for Objects Occurring in Shakespeare's Plays

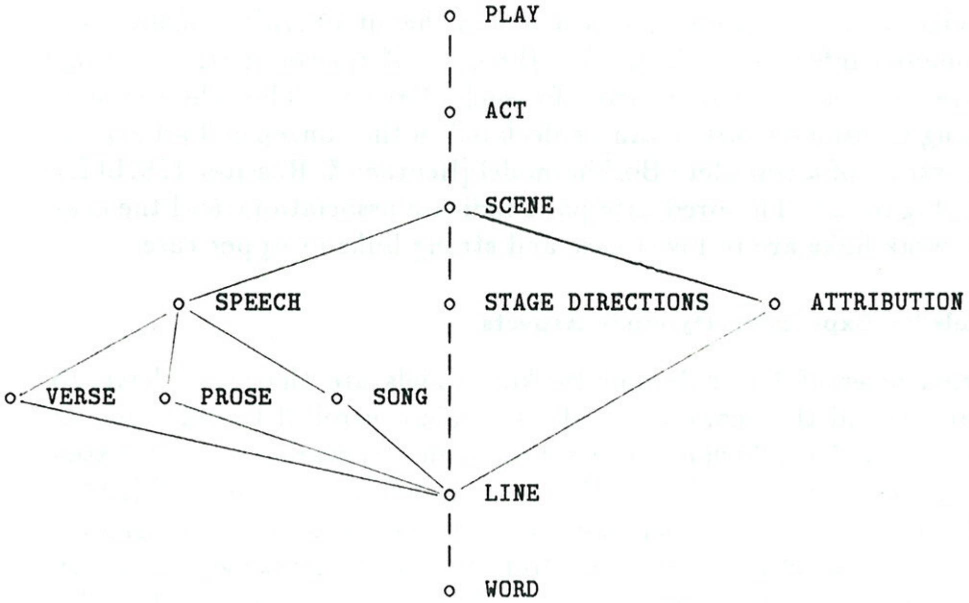


Figure 11. Extract from the Borkin Semantic Model for Legal Text

```
Section
-----------------------------------------------------------------
|in.act:OBJECT      |                 |         in.act:SUBTYPE          |
|contains.part:     | contains.part:|                                 |
|         object    |        subtype|                                 |
|                   | within.part:  |       within.part:subtype       |
|                   |         object |                                 |
|                   |                 |       be section:SUBTYPE        |
-----------------------------------------------------------------
|     act           |     part        |           section               |
-----------------------------------------------------------------
|   force-text      |   force-text    |  force-text  |  whole-text      |
-----------------------------------------------------------------
| year*|chapter*    |   number        |  number*     | section-text     |
-----------------------------------------------------------------

                 ------------------------------------
```

figure is divided into four sections and for economy of presentation the schedule, paragraph and footnote definitions have been excluded. In the first, Figure 12(a), the base class of headings 'heading' is defined consisting of a series of identifiers for text: naturalID for human comprehension, symbolicID for the machine and draftID as a unique identifier for the machine during the drafting process. In defining the other text headings, extensive use is made of the facilities for generalization in Taxis. Thus 'act' and 'part' are both considered to be specializations of 'heading' since they inherit their identifiers from 'heading'.

In Figure 12(b), the components of legal documents holding text are defined. The class 'TextBody' is defined as a specialization of 'Heading' to inherit the identifiers and has the additional properties of 'text' to represent the whole-text and 'xref' to represent cross-references from one text to another. The data-type of 'xref' is 'SymbolicKey' to enforce referential integrity. Classes for 'Section' and 'Subsection' are specializations of 'TextBody' with characteristic properties for section# and subsection#. Thus an entity belonging to the class 'TextBody' can also be considered to belong to the class 'Subsection' if it has a subsection# greater than zero. The base class 'Word' has the single property 'wordValue' to provide a dictionary of all words appearing in the text. As introduced earlier in Figure 7(b), the class 'Word.H1' represents the logical position of word values in the text. It is a specialization of 'Heading' to inherit part of its identifier 'symbolicID' from that class.

The classes for the key structures are shown in Figure 12(c). These show the sets of attributes comprised by each type of identifier referenced in Figure 12(a). The symbolic key class 'SymbolicKey' is represented by a series of integers analagous to those defined earlier in Figure 2. The unique identifier for

Figure 12.
A Taxis-like Specification for Data Structures and Transactions in Drafting Statute Law:
(a) Headings.

```
define AnyDataClass Heading with
  changeable
    naturalID: set of NaturalKey
    symbolicID: set of SymbolicKey
    draftID: set of SymbolicKeyV
  unique
    textID: draftID

define AnyDataClass Part isA Heading with
    part#: {| > 0 |}
    section#: {| 0 |}
    subsection#: {| 0 |}
  changeable
    partHeadings: LooseText
    partSubheadings: LooseText
    crossnotes: ShortText
    footnotesToOldStatutes: ShortText
    textFormattingAttributes: ShortText

define AnyDataClass Act isA Heading with
    chapter: {| > 0 |}
    part#: {| 0 |}
    section#: {| 0 |}
    subsection#: {| 0 |}
  changeable
    title: LooseText
    date: Date
    preamble: ShortText
    arrangements: LooseText
    crossnotes: ShortText
    textFormattingAttributes: ShortText
```

the drafting process of 'SymbolicKeyV' is a specialization of 'SymbolicKey' to inherit the basic identifier with the additional property of 'version#' to differentiate between different versions. The class 'NaturalKey' represents the natural form of the symbolic key for human understanding. The types of the properties are non-integer in these cases to reflect the use of different numbering methods such as by arabic number, roman number, or arabic letter. The class 'NaturalKeyV' posseses multiple inheritance: 'NaturalID' is inherited from the class 'NaturalKey' and 'version#' from 'SymbolicKeyV'.

The production of logical word indexes (concordances) and dynamic aspects of the drafting process can be defined in Taxis by transaction-classes as shown in Figure 12(d). The transaction-class 'WordID' referenced from the class

Figure 12. Taxis-like Specification: (b) Text.

```
define AnyDataClass TextBody isA Heading with
  changeable
    text: WholeText
    xref: set of SymbolicKey

define AnyDataClass Word with
  unique
    wordValue: Word

define AnyDataClass Section isA TextBody with
    section#: {| > 0 |}
    subsection#: {| 0 |}

define AnyDataClass SubSection isA TextBody with
    subsection#: {| > 0 |}

define AnyDataClass Word.H1 isA Heading with
  changeable
    wordValue : set of Word
    position: WordPhysicalPosition
  unique
    wordLogicalPosition: (wordValue, position, symbolicID)
```

'BreakTextToFormWordIndex' produces a succession of values for each text 't' of individual words and their respective physical positions within the text. Each pair of values is inserted in the class 'Word.H1' which automatically inherits the value of 'symbolicID' to record explicitly the logical position of each word in the text. The transaction is not executed unless the prerequisite that the supplied symbolic identifier is an instance of the class 'SymbolicKey' is satisfied.

The transaction AddNewVersionText is a prototype design of a method for handling the dynamic aspects of version management. Prerequisites are defined which must be satisfied before the transaction can be executed: the version number for the text must be new and any cross-references made from the text must satisfy the need for referential integrity. This transaction allows the user to specify the natural key for identification of the text being updated. The mapping between the natural and symbolic keys is handled automatically in the transaction through the transaction-class 'Transform' which converts natural identifiers into their symbolic form.

5.1. 4 Implementations of Semantic Models

It is unfortunate that in spite of their expressiveness, general implementations of the semantic models are not readily available in commercial software [Peckham & Maryanski 1988]. A direct implementation of the E-R model, for

Figure 12. Taxis-like Specification: (c) Keys.

```
define AnyDataClass SymbolicKey  with
  changeable
    subsection#: {| 0:999 |}
    section#: {| 0:999 |}
    part#: {| 0:255 |}
    year: {| 1988:2000 |}
    chapter: {| 0:255 |}
  unique
    symbolicID: (year, chapter, part#, section#, subsection#)

define AnyDataClass SymbolicKeyV isA SymbolicKey with
  changeable
    version#: {| 1:255 |}
  unique
    symbolicIDV: (symbolicID, version#)

define AnyDataClass NaturalKey with
  changeable
    subsection$: ForceText
    section$: ForceText
    part$: ForceText
    act$: ForceText
  unique
    naturalID: (act$, part$, section$, subsection$, version$)

define AnyDataClass NaturalKeyV isA NaturalKey, SymbolicKeyV with
  unique
    naturalIDV: (naturalID, version#)
```

instance, would provide the framework for improved textbase initiatives. By explicitly recording the hierarchical structure of the text as well as providing base relations for set manipulation, this model provides an effective data structuring tool. However, the other important part of a model of operations is undefined so it is not possible to describe the behaviour of objects. Such semantic models as they stand are thus only a partial aid to improved textbase facilities and it may be necessary to use semantic models with control over events for more complete specifications of textbases. Some recent work [Nixon et al 1987] on implementing Taxis is an important step in this direction.

5.2 Basic Models

None of the basic models is rich enough in capability to satisfy all requirements in manipulating complex textual structures. The hierarchical and simple network models can be quickly discounted but a more detailed discussion is necessary to illustrate weaknesses in a relational approach employing 'flattening' of data.

Figure 12. Taxis-like Specification: (d) Transactions.

```
define AnyTransactionClass BreakTextToFormWordIndex
        (s:symbolicID, t:text) with
  locals
  wordFound: Word
  wordPP: WordPhysicalPosition
  prerequisites
    symbolicIDIsValid?   s instanceOf SymbolicKey
  actions
    breakText: wordPP <- 1
               do while wordPP < EOLN
               wordFound <- WordID(t, wordPP)
               insert-object in Word.H1 with (wordValue <- wordFound,
                  position <- wordPP)
               wordPP <- wordPP + 1
               enddo

define AnyTransactionClass WordID ...

define AnyTransactionClass AddNewVersionText
        (nid: naturalID, v:version#, t:text, x:xref) with
  locals
    sid: symbolicID
  prerequisites
    versionDoesNotExist? (nid, v) notInstanceOf NaturalKeyV
    xrefIsValid?   x instanceOf SymbolicKey
  actions
    addNewVersion: insert-object in NaturalKey with naturalID <- nid
                   insert-object in NaturalKeyV with version# <- v
                   sid <- transform(nID)
                   insert-object in SymbolicKey with symbolicID <- sid
                   insert-object in SymbolicKeyV with version# <- v
                   insert-object in TextBody with text <- t

define AnyTransactionClass Transform ...
```

Pure hierarchical systems are inadequate for complex data structures such as those found in textual and administrative data because they only allow the modelling of one-to-many relationships. Without object-oriented property inheritance and with only intra-record hierarchical structures available for linking data together, record sizes become very large when for instance one act of the statute law occupies over 0.5Mb.

The network model is more general than the hierarchical model for it allows the modelling of many-to-many relationships. A large number of implementations exist and it has proved applicable in administrative and commercial areas. To implement, however, a successful application of the network model needs the involvement of professional programming staff even for record retrieval because of the intimate knowledge that is required of the physical structure. In particular, network query languages are not suited to iterative searching and this may be the chief reason why the network model has not been used extensively for text.

5.2. 1 The Relational Model

The relational model [Codd 1970] on the other hand offers the power of the network model but with a simple and elegant method of data manipulation. Relational systems have been used with success in administrative areas where fixed units are adequate although Codd in 1979 [Codd 1979] suggested that even here variable units using a supertype-subtype approach would be useful.

The E-R model given earlier for law can be implemented directly by mapping the table-types in Figure 7(b) on to conventional relations, each table-type being represented by a relation. However, the semantics of the E-R diagram are now represented implicitly; for instance, the structure of the objects:

$$act- > schedule- > paragraph- > subparagraph- > word$$

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. To a user with detailed knowledge of relational data structures and manipulation, a clear advantage of the relational model is the ease with which keys in the form of formatted data can be manipulated giving flexibility in the choice of units for display purposes and logical addressing on symbolic keys for navigation purposes. Thus the use of standard set operators can provide aggregation of data, dynamic variation of the unit of retrieval as users' needs change and cross-referencing [Rossiter 1986]. To a user with little knowledge of relational structures, the disadvantage of the relational approach is the dependence on views predefined as a series of external schemata by the database administrator to achieve abstractions such as aggregation.

Of equal importance, not only with most current implementations but also in developing flexible systems for the future, is the problem of indexing text in a versatile manner, Word searching can be performed sequentially on any base or aggregated unit but for speed of access in large textbases, it is essential to be able to locate rapidly the exact positions of words in the text. Word indexes are thus essential for most text applications. Relational systems provide indexing facilities for simple data but are not too suited to the indexing of complex data. There is no real concept of a 'global' index in relational databases to hold values from more than one relation: 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. Thus, in Figure 7(b), it is not easy to build a single index on the abstraction 'text'. When it is considered that this index should also be of data-type 'word' and allowance is to be made for context and proximity matching, the difficulties are increased.

5.2. 2 Extended Relational Approach

The logical approach to improving the flexibility of relational systems in manipulating text is to flatten the data so that it is completely normalized down to the word level. In principle, the existence of relations holding words, together with their positions in the text, and of the base relations shown in Figure 7(b) provides data structures which can satisfy most user requirements. However, as indicated earlier, some features of Figure 7(b) are not easy to represent in a relational implementation. Thus the key attributes 'text.id', 'citing.text.id' and 'cited.text.id' which are essential for generalization need to be specified in an explicit manner that conforms to third normal form and entity integrity.

In the law example, five logical word indexes are needed in the form of the fully-fledged relations shown in Figure 13. This number of word indexes is required because the complex object 'act' comprises five distinct paths to 'word' via 'footnote', 'subsection', 'section', 'subparagraph' and 'paragraph' respectively. In free text retrieval systems, the requirement for separate paths is generally glossed over by overlaying the different paths onto the single hierarchical structure shown in Figure 6, although there are exceptions such as Basis [Bain et al 1989] in which the use of indexing prefixes enables multiple hierarchies to be contrived. Figure 13 also defines the relation 'request' which contains the words in the user's search request and could also include other relation definitions for data such as thesauri and stop and go lists so that all data can be declared in fully fledged relations.

The word indexes allow searches on words to be quickly satisfied. Further, they enable the unit searched to be varied dynamically by taking views of the index using the projection operator and dividing the viewed relation by the relation 'request' which contains the search terms. 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.H5[YEAR, CHAPTER, SCHEDULE#, WORD] divideby REQUEST[WORD])
UNION
(WORD.H6[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

Figure 13. Relations holding Logical Word Indexes for the Paths in the Text Structures

```
 Relations: (attributes comprising the primary key are italicised)
 Word.H2 ( position, footnote#, year, chapter, word)
 Word.H3 ( position, ss#, section#, year, chapter, word)
 Word.H4 ( position, section#, year, chapter, word)
 Word.H5 ( position, subp#, para#, schedule#, year, chapter, word)
 Word.H6 ( position, para#, schedule#, year, chapter, word)

 Request( word)

 Paths:
H2            H3              H4            H5              H6
|             |               |             |               |
|             |               |             |               |
o  ACT        o  ACT          o  ACT        o  ACT          o  ACT
|             |               |             |               |
|             |               |             |               |
|             o  SECTION      |             o  SCHEDULE     o  SCHEDULE
|             |               |             |               |
|             |               |             |               |
o  FOOTNOTE   |               o  SECTION    o  PARAGRAPH    |
|             |               |             |               |
|             o  SUBSECTION   |             |               o  PARAGRAPH
|             |               |             |               |
|             |               |             o  SUBPARAGRAPH |
|             |               |             |               |
|             |               |             |               |
o  WORD       o  WORD         o  WORD       o  WORD         o  WORD
```

effort and would be likely to result in inefficient code. Stonebraker [Stonebraker 1986] considers some of the extra facilities that are required by relational systems to be effective in logical indexing. These include a BREAK operator, for creating a new relation to hold the words comprising a document and their position within it, and a CONCAT operator to rebuild documents from word indexes. Such facilities could also be provided by a standard programming language such as SNOBOL using a combination of the BREAK and SPAN patterns.

In an earlier paper [Heather & Rossiter 1989a], we considered in detail the manipulation of text in relational databases using the example of the Bible. In that study, it was found feasible to normalize textual data to at least the word level and to employ SQL or relational algebra for searching the data against different unit sizes and for aggregating the data as necessary. The biblical data were represented by a complex object with the single path shown in Figure 14 which is very much simpler to manage than the multiple paths of law shown in Figure 13. However, even with the biblical example, the approach of 'flattening' the data would be cumbersome for data manipulation, would hide the natural structure of the data from the user and have adverse performance implications when reconstituting aggregations for documents in large textbases. With the more complex objects, the multiple paths and shared sub-objects found in the law and Shakespeare, it is not clear how an acceptably high level of data abstraction can be presented to the user while providing the facilities with standard set operators. A more detailed description of the fundamental deficiencies of the relational model for handling text is given in a technical report in preparation by the present authors [Heather & Rossiter 1989c].

Figure 14. Class Structure for Objects Occurring in Biblical Text

```
o  TESTAMENT
|
|
o  BOOK
|
|
o  CHAPTER
|
|
o  VERSE
|
|
o  WORD
```

6: Object Oriented Systems

In advanced high-level languages such as Ada 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 (for example, Simula) and to achieve communication between objects by messaging (for example, Ada). These systems, often now called object-oriented, thus readily allow objects to be built from other objects, iterative searching of complex objects, multiple levels of abstraction and a natural ability to handle dynamic aspects including integration in textual analysis of function with data, all important issues for the handling of 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, Chisholm & Cockshott 1981]. One of the earliest developments was GemStone [Copeland & Maier 1984] which is built on SmallTalk and uses the Opal language for data definition and manipulation. Abstract data types can be defined, object identity is preserved independent of value and objects participate in one or more collections to provide a shared subobject facility. Behavioural aspects are handled by messaging. Another early system is Vbase [Andrews & Harris 1987] which provides the ability to define super-types, methods and procedures. In relational database design, it is claimed that object-oriented techniques naturally provide the required level of abstraction [Blaha, Premerlani & Rumbaugh 1988].

The strengths of the object-oriented approach lie in the ability to import advanced programming techniques into areas of data modelling where 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. For example, Maier [Maier 1986] has investigated the indexing of different paths through a complex object such as that presented earlier in Figure 13.

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 users usually have a relatively sophisticated programming background. However, in areas such as text, office automation and CASE, 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.

Other work has centred on extending the relational model in an object-oriented manner. Stonebraker has considered how relational systems can be extended to be applied to a wider range of applications. In Postgres [Stonebraker, Anton & Hanson 1987], abstract data types with property inheritance can be defined by the user and attributes can be of type 'procedure' calling code written by the user in C or QUEL. This system also provides rules which enable constraints to be readily applied and time-stamps all data as a first step towards version management. One of the problems with the procedures is that, on execution, multiple values may be returned which cannot be placed in a relation directly so that the property of closure is lost. Further, the reliance on procedures to achieve some abstractions rather than on higher-level modelling constructions offends end-users who require non-procedural access. A similar development can be seen with Generis [McInnes 1989] which builds a class structure and rule system on top of the relational model. This provides near-natural language and menu interfaces so that users can be given a cleaner interface than with many other object-oriented approaches including the elimination of logical navigation. However, the efficacy of the natural language approach must be open to some doubt. For updating, it may be too uncertain for its use to be justified as incorrect parsing could produce unintended changes to data. In retrieval, users may obtain a response that looked plausible but was in fact only partially correct. For certainty of retrieval and update, users may need to revert to an alternative procedural interface.

7: Discussion

In contrast to Figure 5 which illustrates textbase requirements from the users' perspective, Figure 15 summarizes the required facilities for textbases from the system viewpoint and shows the extent to which they are met by the various techniques of free text retrieval, ISO-standard relational database, extended relational database with facilities to flatten data, semantic models oriented towards static and dynamic aspects such as E-R and Taxis respectively, and object-oriented systems.

Examination of Figure 15 shows that free text 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. However, aggregation is a cumbersome task for a user and dynamic behaviour is not considered. The E-R model has not been generally implemented so the information in the figure has to be incomplete but with the lack of defined operations there appears to be no real basis for a complete answer to users' problems.

The object-oriented approaches appear to offer the most promise whether in the guise of semantic models like Taxis or databases such as GemStone. Such systems handle quite naturally variable unit size, shared subobjects, dynamic behaviour and integration of function and data, and could provide the basis for an initiative in multi-media modelling. However, so far, object-oriented systems have presented relatively procedural interfaces to users, are rather limited in standard database functions such as providing 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

Figure 15. Requirements for Textbases from System Viewpoint

	Free Text	Relational		Semantic		Object-oriented
		stand.	extend.	E-R extend.	Taxis	
1. Structures						
. unlimited data size	yes	not so far	not so far	-	-	-
. symbolic id	no	yes	yes	yes	yes	yes
. aggregation (inter-object)	no	SQL+ views	SQL+ views	no	yes	yes
. generalization	no	no	no	yes	yes	yes
. hierarchical	yes	yes	yes	yes	yes	yes
. shared sub-objects	no	no	with difficulty	yes	yes	yes
3. Retrieval						
. non-procedural (relatively)	yes	in part	in part	-	no	no
. fast	yes	only on base	yes	-	-	-
. context	yes	yes	yes	-	yes	yes
. proximity	yes	no	yes	-	yes	yes
. formatted data	limited	yes	yes	-	yes	yes
4. Navigation	no	yes	yes	yes	yes	yes
6. Temporal						
. in-place update	yes	yes	yes	-	yes	yes
. dynamic behaviour	no	no	no	no	yes	yes
. concurrent access	yes	yes	yes	-	limited	limited
. value inheritance	no	no	no	yes	yes	yes
7. Views	no	yes	yes	-	yes	yes
10. Analysis						
. function integrated with kernel	no	no	no	-	yes	yes
11. Multi-media						
. unified model	no	no	possible	possible	possible	possible

Note: '-' indicates no information available.

thought of as object-bases [Heather 1989] 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 & Nierstrasz 1988] and much research of a fundamental nature is still required to attain a single complete multi-media model.

8: References

[Andrews & Harris 1987] Andrews, T, & Harris, C, (1987), Combining Language and Database Advances in an Object-oriented Development Environment, OOPSLA '87 Conference Proceedings, ACM SIGPLAN Notices **22**(12) 430-440.

[Atkinson, Chisholm & Cockshott 1981] Atkinson, M P, Chisholm, K J, & Cockshott, W P, (July 1981), PS-Algol: an Algol with a persistent heap, ACM SIGPLAN Notices **17**(7); also available as Technical Report CSR-94-81, Computer Science Report, Edinburgh University.

[Bain et al 1989] Bain, M, Bland, R, Burnard, L, Duke, J, Edwards, C, Lindsey, D, Rossiter, N, & Willett, P, (1989), Free Text Retrieval Systems, A Report and Evaluation, Taylor Graham, London.

[Blair & Maron 1985] Blair, D C, & Maron, M E, (1985), An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System, CACM **28** 289-299.

[Blaha, Premerlani & Rumbaugh 1988] Blaha, M R, Premerlani, W J, & Rumbaugh, J E, (1988), Relational Database Design: Using an Object-oriented Methodology, CACM **31**(4) 414-427.

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

[Borkin 1979] Borkin, S A, (1979), Equivalence properties of semantic data bases for database systems, Technical Report of Laboratory for Computing Science, MIT, TR-206.

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

[Codd 1970] Codd, E F, (1970), A Relational Model of Data for Large Shared Data Banks, CACM **13**(6) 377-387.

[Codd 1979] Codd, E F, (1979), Extending the Database Relational Model to capture more meaning, ACM TODS **4** 397-434.

[Connolly 1985] Connolly, R, (1985), The statute law on property database: an implementation of a full text data base on a relational data base management system, M.Sc. Dissertation, Computing Laboratory, University of Newcastle upon Tyne, D313.

[Copeland & Maier 1984] Copeland, G, & Maier, D, (1984), Making SmallTalk a Database System, Proc ACM/SIGMOD International Conference on the Management of Data **14**(2) 316-324.

[Dadam et al 1987] Dadam, P, Kuespert, K, Andersen, F, Blanken, H, Erbe, R, Guenauer, J, Lum, V, Pister, P, & Walch, G, (1987), A DBMS Prototype to support Extended NF2 Relations: An Integrated View on Flat Tables and Hierarchies, ACM SIGMOD 356-367.

[Davis 1983] Davis, D, (1983), Keyword analysis of legal text, M.Sc. Dissertation, Computing Laboratory, University of Newcastle upon Tyne D245.

[Diamandis 1986] Diamandis, I, (1986), An electronic filing system for an office environment using SPIRES, M.Sc. Dissertation, Computing Laboratory, University of Newcastle upon Tyne, D327.

[Eakins 1989] Eakins, J, (1989), SAFARI: A Shape Retrieval System for Engineering Drawings, 11th Colloquium on Information Retrieval Research, Huddersfield Polytechnic.

[du Feu 1977] du Feu, D, (1977), A Computer-based Welfare Benefits Information System, in: System Documentation, IBM UK, Scientific Centre, Peterlee; Hafner, C D, (1981), An Information Retrieval System based on a Computer Model of Legal Knowledge, Ann Arbor.

[Earl et al 1986] Earl, A N, Whittington, R P, Hitchcock, P, & Hall, A, (1986), Specifying a Semantic Model for use in an Integrated Project Support Environment, in: Software Engineering Environments, Sommerville, I, (ed), IEE Computing Series 7, London.

[Heather 1986] Heather, M A, (1986), Future Generation Systems in the Service of the Law, in: Automated Analysis of Legal Texts, eds. Martino, A A, Socci Natali, F, North-Holland, Amsterdam 643-660.

[Heather 1989] Heather, M A, (May 1989), Law as a Knowledge-Object Base, International Conference on Law & Artificial Intelligence, University of Bologna, Italy.

[Heather & Rossiter 1987a] Heather, M A, & Rossiter, B N, (1987), Database techniques for text modelling: the document architecture of British statutes, University of Newcastle upon Tyne, Computing Laboratory Technical Report no 227.

[Heather & Rossiter 1987b] Heather, M A, & Rossiter, B N, (1987), The Textual Environment and Database Management Systems, in: Empirical Foundations of Information and Software Science III, edd. Rasmussen, J, & Zunde, P, Plenum, New York 45-62.

[Heather & Rossiter 1988] Heather, M A, & Rossiter, B N, (1988), Specialist Dictionaries in Electronic Form, Literary & Linguistic Computing 3(2) 109-121.

[Heather & Rossiter 1989a] Heather, M A, & Rossiter, B N, (1989), Syntactical Relations in Parallel Text, in: Proceedings 15th International ALLC Conference, Choueka, Y, (ed.), Jerusalem 1988.

[Heather & Rossiter 1989b] Heather, M A, & Rossiter, B N, (1989), A Generalized Database Management Approach to Textual Analysis, in: Proceedings 2nd International Colloquium, Bible and Computer: Methods, Tools, Results, Champion-Slatkine, Paris-Geneva 519-536.

[Heather & Rossiter 1989c] Heather, M A, & Rossiter, B N, (1989), Managing General Data in Business Information Systems: Mediating the Relational Model for Text in the Legal Office, University of Newcastle upon Tyne, Computing Laboratory Technical Report (in prep.)

[Howe 1983] Howe, D R, (1983), Data Analysis for Data Base Design, Edward Arnold, London 126-167.

[Hudson 1985] Hudson, G, (1985), Establishment of a data base containing the information of Hansard using a hierarchical data base management system SPIRES, M.Sc. Dissertation, Computing Laboratory, University of Newcastle upon Tyne, D289.

[Kemper & Lockemann (1987] Kemper, A, & Lockemann, W M, (1987), Object-oriented Database System for Engineering Applications, Proceedings ACM Special Interest Group on Management of Data, 1987 Annual Conference, Dayal, U, & Traiger, I, (edd.), SIGMOD Record **16**(3) 299-310.

[Linton 1982] Linton, A S J, (1982), The use of collocates as an aid to text retrieval, M.Sc. Dissertation, Computing Laboratory, University of Newcastle upon Tyne, D226.

[Lorie 1981] Lorie, R A, (1981), Issues in databases for design application, IBM Research Report RJ3176.

[Lorie et al 1985] Lorie, R, Kim, W, McNabb, D, Plouffe, W, & Meier, A, (1985), Supporting Complex Objects in a Relational System for Engineering Databases, in: Query Processing in Database Systems, Kim, W, Reiner, D S, & Batory, D S, (edd.) 145-155, Springer-Verlag, New York.

[Kowalski & Sergot 1987] Kowalski, R, & Sergot, M, (1987), Leith and Legal Logic Programming, Computer Journal **30**(3) 285; Leith, P, (1988), Legal Logic Programming, Computer Journal **31**(1) 92-93.

[Leith 1986] Leith, P, (1986), Fundamental Errors in Legal Logic Programming, Computer Journal **29**(6) 545-552.

[Maier 1986] Maier, D, (1986), Indexing in an Object-Oriented DBMS, Proceedings International Workshop on Object-Oriented Database Systems, IEEE 171-182.

[McInnes 1989] McInnes, S, (1989), The Generic Relational Data Model: A new Framework for Knowledge Representation, Deductive Systems Ltd, Technical Report 01-89.

[Mitchell 1989] Mitchell, J, (1989), Implementation of a Relational Textbase, Dissertation submitted for M.Sc., Computing Laboratory, University of Newcastle upon Tyne.

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

[Nelson 1988] Nelson, T H, (Jan 1988), Managing Immense Storage, Byte 225-238.

[Newton 1981] Newton, J E, (1981), A "user friendly" interface for a law data base retrieval system, M.Sc. Dissertation, Computing Laboratory, University of Newcastle upon Tyne D212.

[Nixon et al 1987] Nixon, B, Chung, L, Lauzon, D, Borgida, A, Mylopoulos, J, & Stanley, M, (1987), Implementation of a Compiler for a Semantic Data Model: Experience with Taxis, ACM SIGMOD 118-131.

[Peckham & Maryanski 1988] Peckham, J, & Maryanski, F, (1988), Semantic Data Models, ACM Computing Surveys **20**(3) 153-189.

[Raymond & Tompa 1988] Raymond, D R, & Tompa, F W M, (1988), Hypertext and the Oxford English Dictionary, CACM **31**(7) 871-879.

[Rossiter 1986] Rossiter, B N, (1986), Full Text Database Management Systems: A Model and Implementation for Law, in: Automated Analysis of Legal Texts, edd. Martino, A A, & Socci Natali, F, North-Holland, Amsterdam 899-916.

[Rossiter 1987] Rossiter, B N, (1987), Machine Awareness in Database Technology, Proceedings Symposium VI, Meta-intelligence and the Cybernetics of Consciousness. XI International Congress of Cybernetics, Namur 1-9.

[Rossiter et al 1988] Rossiter, B N, Davis, P, Goodman, D S G, Ward, M K, & Heather, M A, (1988), Generalised DBMS as a tool for research, University Computing **10**(2) 71-79 & **11**(2) 116.

[Sakai 1983] Sakai, H, (1983), Entity Relationship Approach to Logical Data base Design, in: Entity Relationship Approach to Software Engineering, North-Holland 155-187.

[Salton 1986] Salton, G, (1986), Another Look at Automatic Text-retrieval Systems, CACM **29** 648-656.

[Sergot et al 1986] Sergot, M J, Sadri, F, Kowalski, R A, Kriwaczek, F, Hammond, P, & Cory, H T, (1986), The British Nationality Act as a Logic Program, CACM **29** 370-386.

[**Smith & Smith 1977**] Smith, J, & Smith, D, (1977), Data Abstraction, Aggregation and Generalization, ACM TODS **2**(2) 105-133.

[**Sparck-Jones & Wilks 1985**] Sparck-Jones, K, & Wilks, W, (1985), Automatic Natural Language Parsing, Ellis Horwood.

[**Stonebraker 1986**] Stonebraker, M, (1986), Document Processing in a Relational Data base System, The INGRES Papers, Addison-Wesley 357-375.

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

[**Tsichritzis & Klug 1978**] Tsichritzis, D C, & Klug, A, (edd), (1978), ANSI X3/SPARC DBMS Framework, Report of the Study Group on Data Base Management Systems, Information Systems 3.

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

[**Walter 1989**] Walter, C, (1989), Legal Abstractions in PLEX, a Legal Expert System for Determining the Validity of Patents, International Conference on Law & Artificial Intelligence, University of Bologna, Italy.

[**Wilson 1988**] Wilson, E, (1988), Justus: towards a Workstation for Information Retrieval in Law, Preproceedings 4th International Congress on Law and Computers, Session X, Italian Ministry of Justice, Rome.

[**Zaniolo et al 1986**] Zaniolo, C, Ait-Kaci, H, Beech, D, Cammarata, S, Kerschberg, L, & Maier, D, (1986), Object Oriented Database Systems and Knowledge Systems, in: Kerschberg, L (ed), Expert Database Systems: Proceedings from the First International Workshop, Benjamin Cummings, Menlo Park.