and in the same schedule by:

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

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

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

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

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

5 Object Oriented Systems

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

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