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

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

4.2 Extended Relational Approach

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

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

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

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