A STORAGE MANAGER FOR THE HYPERNODE MODEL

E. Tuv, A. Poulovassilis and M. Levene

1,3 Department of Computer Science, University College London, Gower St., London WC1E 6BT e-mail: {e.tuv, m.levene}@uk.ac.cs.ucl

2 Department of Computing, King's College London, Strand, London WC2R 2LS e-mail: a.poulovassilis@uk.ac.kcl.cc.oak

Abstract

We describe the implementation of a Storage Manager (SM) for the Hypernode model, a new data model whose aim is to integrate object-oriented and deductive databases. The single data structure of this model is the hypernode, a directed graph whose nodes may themselves be directed graphs. The components of the SM manipulate these graphs in a persistent store. The main effort of the first prototype of the SM has been to develop a modular and extensible system which can be used as a reliable and stable core for future versions. In particular, the SM caters for object-identity and referential sharing between hypernodes, large and dynamic hypernodes, clustering strategies on secondary storage, and retrieval operations which utilise indexing techniques. The main contribution of the SM is the single graph data structure which permeates throughout all the levels of the implementation; in this way efficiency can be achieved within all the components of the SM as a result of optimising this data structure, and also interfacing between the components of the SM is simple and uniform.

1 Introduction

Recent database research has focussed on deductive and object-oriented databases [ULL88, GAR89]. These are largely complementary: the former supports both stored and derived relations and the latter supports data abstraction mechanisms such as classification, identification, encapsulation and inheritance. Hence, recent database research has aimed at integrating the two paradigms. This integration has generally taken the route of extending logic-based deductive database languages with features such as object identity, sets, functions, methods and inheritance [ULL91]. Taking a different approach, we have developed a graph-based model called the Hypernode Model [LEV90, POU90]
to support such an integration. In contrast to other graph-based data models \[CON90, GYS90, LEV91, TOM89\], we use nested, possibly recursively defined, directed graphs termed hypernodes.

A hypernode is a pair, \((N,E)\), of nodes and directed edges such that the nodes of \(N\) are either primitive values or themselves hypernodes. Hypernodes have unique value-independent labels which serve as object identifiers. We illustrate a hypernode in Figure 1. It represents a couple, \(C\), consisting of two people, \(PER1\) and \(PER2\), whose children are nested within further hypernodes. In Figure 2 we show the children of person \(PER2\), which would become visible if we “exposed” the hypernodes labelled \(PER3\) and \(PER4\).

The labels \(C\) and \(PER1-PER4\) in these figures are superscripted with the tags \(COUPLE\) and \(PERSON\), respectively. These tags indicate the types of their associated hypernodes and are omitted whenever they are understood from context. Types give us a means of defining database schemas and of enforcing constraints on the structure and content of hypernodes (see \[POU90\] for a detailed description of types). In fact, types are just hypernodes and so can be queried and updated using the same formalism as for data. We also note the use of the node \(none^{PERSON}\) in Figure 2 - it is null value of type \(PERSON\) denoting “does not exist”.

The hypernode model comes equipped with a computationally powerful declarative language called Hyperlog. Thus, the model and language share features with both