PRIMA - A Database System Supporting Dynamically Defined Composite Objects

Michael Gesmann, Andreas Grasnickel, Theo Härder, Christoph Hübel,
Wolfgang Käfer, Bernhard Mitschang, Harald Schöning
Sonderforschungsbereich 124, University Kaiserslautern, P.O. Box 3049
W-6750 Kaiserslautern, Germany
{gesmann, grasnick, haerder, huebel, kaefer, mitsch, schoenin}@informatik.uni-kl.de

PRIMA [1] is a non-standard database system developed at the University Kaiserslautern. Its major purpose is the support of engineering design applications, such as VLSI design and software engineering. The applications require tailored application-dependent interfaces which, however, all share basic notions like that of a composite object. Hence, the approach of PRIMA is to offer an application-independent complex-object interface (the molecule-atom data model, shortly called MAD model [3]) and to provide means to easily augment this interface by application-dependent functionality. In the following, we will concentrate on the MAD model and its implementation.

The MAD model allows for the dynamic definition of complex object types (called *molecule types*) at query time. These molecule types together with the database induce sets of molecules which can be retrieved and manipulated by the MAD model's query language MQL. Molecules are built from *atoms*, which are stored in the database. Atoms are interconnected by bidirectional links, thus forming an undirected network of atoms. The molecule type definition specifies a template which is applied to this network, forming directed subgraphs (molecules) thereof. Each molecule is a coherent directed graph with exactly one root (root atom type). Molecules may have a hierarchical, a network-like, or a recursive structure [5]. They may be disjoint or not-disjoint. This molecule definition facility is a basic building block to formulate composite object queries which, in turn, define molecule types that can be used in other queries (closure of the data model [4]).

The MQL query is structured in accordance to the SQL language. The molecule type definition is done in the FROM clause. A graph-defining notation is used, consisting of "-" which means *follow a link between the building blocks on both sides of the "-" in left-to-right direction*, and parentheses and commas to express branches and intersections. The building blocks may consist of atom types, predefined molecule types, or MQL queries. For example, the molecule type definition A-(B,C) means: for each atom of type A, follow all links leading to atoms of type B or type C. The result is a set of directed graphs, one for each atom of type A existing in the database. Each graph contains one atom of type A, and all atoms of types B and C which are connected to this A atom in the database. The molecule set defined in this way may be restricted by a condition in the WHERE clause. A projection may be specified in the SELECT clause.

The implementation of PRIMA is designed to exploit coarse-grain parallelism [2]. For this purpose, PRIMA consists of a static set of multi-tasking processes, each possibly residing on another processor. Processes use shared memory for communication whenever possible, otherwise they employ a message passing mechanism. Obviously, this concept leaves a lot of freedom in configuring the system. One has to decide, how many processes for each functionality shall be used, and how they shall be distributed over the available processors.

In the video, we illustrate some main features of the MQL retrieval facilities. Furthermore, we sketch the implementation of the MAD model. Finally, we present an analysis tool which allows for monitoring the behavior of our distributed system. This tool is used to evaluate configuration decisions and as well as to demonstrate the dynamics of the overall system.

References

[1] Härder, T., Meyer-Wegener, K., Mitschang, B., Sikeler, A.: PRIMA - A DBMS Prototype Supporting Engineering Applications, in: Proc. 13th VLDB, Brighton, 1987, pp. 433-442.

[2] Härder, T., Schöning, H., Sikeler, A.: Parallelism in Processing Queries on Complex Objects, in: Proc. Int. Symp. on Databases in Parallel and Distributed Computing, Austin, TX, 1988, pp. 131-143.

[3] Mitschang, B.: Towards a Unified View of Design Data and Knowledge Representation, in: Proc. 2nd Int. Conf. on Expert Database Systems (EDS), 1988, pp. 33-50.

[4] Mitschang, B.: Extending the Relational Algebra to Capture Complex Objects, in: Proc. 15th VLDB, Amsterdam, 1989, pp. 297-305.

[5] Schöning, H.: Integrating Complex Objects and Recursion, in: Proc. 1st Int. Conf. on Deductive and Object-Oriented database Systems (DOOD), Kyoto, 1989, pp. 535-554.