# PRISMA Database Machine: A Distributed, Main-Memory Approach *

Research Issues and a Preliminary Architecture

Peter M.G. Apers at the University of Twente
Martin L. Kersten at the Centre for Mathematics and Computer Science
Hans C.M. Oerlemans at Philips Research Laboratories Eindhoven

**Abstract**

The PRISMA project is a large-scale research effort in the design and implementation of a highly parallel machine for data and knowledge processing. The PRISMA database machine is a distributed, main-memory database management system implemented in an object-oriented language that runs on top of a multi-computer system. A prototype that is envisioned consists of 64 processing elements.

## 1 PRISMA Project

The long term objective of the PRISMA project is to obtain a flexible architecture for a machine that effectively stores and manipulates both data and knowledge. This long term objective translates into the following goals:

- The construction of a multi-computer system, consisting of a large number of processing elements connected via a message-passing network;
- The definition and efficient implementation of a parallel object-oriented language, called POOL-X;
- The design and implementation of a main-memory database system in POOL-X;
- The design and implementation of an expert system shell in POOL-X that exploits parallelism for inferencing;
- The investigation of using medium to coarse grain parallelism for data and knowledge processing applications, the integration of data and knowledge processing, and the evaluation of the prototype multi-computer, among other things.

The project started in October 1986 and is scheduled until September 1990. The project team consists of members of the Philips Research Laboratories in Eindhoven and the following Dutch academia: Centre for Mathematics and Computer Science, University of Amsterdam, University of Leiden, University of Twente, University of Utrecht. Currently approximately 30 people are directly involved.

*The work reported in this document was conducted as part of the PRISMA project, that is partially supported by the Dutch "Stimuleringsprojectteam Informaticaonderzoek (SPIN)."

# 2 PRISMA Database

## 2.1 Key Issues in the PRISMA Database Machine

The most important ideas behind the design of the PRISMA database machine are summarized below:

- it aims at performance improvement by introduction of parallelism and by using a very large main-memory as primary storage;
- it is designed as a tightly coupled distributed database system;
- it provides an SQL and a logic programming interface;
- it uses a knowledge-based approach to exploit parallelism;
- it uses a generative approach for data managers;

A comparison with other systems can be found in [1].

## 2.2 Global Architecture

The PRISMA DBMS consists of centralized database systems, called One-Fragment Managers (or OFM), running under the supervision of a Global Data Handler (or GDH). The architecture of an OFM is explained in section 2.5. The GDH contains the data dictionary, the query optimizer, the transaction manager, the concurrency control unit, and the parsers for SQL and PRISMAlog, a logic programming language like Datalog. Besides these components, there is a recovery component and a data allocation manager.

Parallelism will be used both within the DBMS and in query processing. Within the DBMS this will be obtained by running several instances of components of the DBMS in parallel. Examples of these components are the parsers, the query optimizer, the transaction monitor, and the OFMs for intermediate results. For each query a new instance is created, possibly running at its own processor. This means that evaluation of several queries and updates can be done in parallel, except for accesses to the same copy of base fragments of the database.

## 2.3 PRISMAlog

The logic programming language that is defined in PRISMA is called PRISMAlog and has an expressive power similar to Datalog and LDL. It is based on definite, function-free Horn clauses and its syntax is similar to Prolog. One of the main differences between pure Prolog and PRISMAlog is that the latter is set-oriented, which makes it more suitable for parallel evaluation.

The semantics of PRISMAlog is defined in terms of extensions of the relational algebra. Facts correspond to tuples in relations in the database. Rules are view definitions including recursion.

## 2.4 Query Optimization

To exploit parallelism, on a large pool of processors, is a non-trivial task. One dimension of the problem is the grain size of parallelism: coarse or fine grain. In the PRISMA-project the coarse grain approach is taken, because we expect to gain more performance from it in our multi-computer architecture.

A knowledge-based approach to query optimization is chosen to exploit all this parallelism in a coherent way. The knowledge base contains rules concerning logical transformations, estimating

sizes of intermediate results, detection of common subexpressions, and applying parallelism to minimize response time.

### 2.5 Architecture of One-Fragment Managers

The DBMS software is organized as a fully distribibuted database system in which the components are, so-called, One-Fragment Managers (or OFM). These OFMs are customized database systems that manage a single relation fragment. They contain all functions encountered in a full-blown DBMS; such as local query optimizer, transaction management, markings and cursor maintenance, and (various) storage structures. More specifically, they support a transitive closure operator for dealing with recursive queries.

Several OFM types are envisioned, each equiped with the right amount of tools. For example, OFMs needed for query processing only, do not require extensive crash recovery facilities. Moreover, each OFM is equiped with an expression compiler to generate routines dynamically. In this way the architecture of an OFM is tuned towards the requirements that can be derived from the relation definition and it avoids the otherwise excessive interpretation overhead incurred by a query expression interpreter.

## 3 PRISMA Machine

The PRISMA machine implements a parallel object-oriented language (POOL-X) on a multi-computer system.

### 3.1 POOL-X

The programming model of POOL-X is a collection of dynamically created processes. Internally the processes have a control flow behaviour and they communicate via message-passing only, i.e. no shared memory. The computational grain of parallelism is medium to coarse.

POOL-X is closely related to POOL2 [2] a general purpose object oriented language. It is strongly typed and hides the multi-computer details. POOL-X is somewhat tailored to the data- and knowledge-processing applications. At a few specific points it introduces dynamic typing to efficiently support the implementation of relation types. Furthermore POOL-X supports explicit allocation of the dynamically created processes onto processing elements. This allows for a proper balance between storage, processing, and communication, under the control of the implementor of the database system.

### 3.2 Multi-computer architecture

The multi-computer architecture envisaged in PRISMA encompasses a number of processing elements connected via a high-bandwidth message-passing network. All processing elements have (data)processing, communication and (local) storage capabilities. The various capabilities have to be balanced in such a way that ultimately the processing elements can be integrated in VLSI. In this way a powerful multi-computer can be implemented cost effectively, by simple replication of (cheap) components.

In a first (prototype) setup the multi-computer will consist of 64 processing elements. Each processing element will have four communication links running at 10 Mbit/sec, and a local main-memory of 16 MByte. The topology of the interconnection network will be mesh-like or a variant of a chordal ring [2]. Various simulations show an average network throughput of upto 20.000 packets (of 256 bits) per second for each processing element simultaneously. Given the high communication bandwidth between processing elements, central resource management is feasible

and site autonomy problems as appearing in current distributed database management systems do not have to be considered.

Apart from the local main-memory, some of the processing elements will also be connected to secondary storage (disk). Using these, the multi-computer system implements stable storage and automatic recovery upon system failures. This approach leads to a simplification in the design of the database management system.

# 4 Current Status and Future Plans

Currently the main components of PRISMA, viz. an expert system, database management system and multi-computer, are identified and defined. In the next three years prototypes will be designed and implemented. A prototype implementation of the DBMS is envisioned to be running on an implementation of POOL-X on a sequential computer in the summer of 1988. Subsequently more extended prototypes on the actual PRISMA multi-computer will be realized.

# 5 Acknowledgement

It is a pleasure to acknowledge the contributions of all PRISMA project members involved; the work described in this document is the result of a stimulating cooperation.

# References

[1] M.L. Kersten, P.M.G. Apers, M.A.W. Houtsma, H.J.A. van Kuijk, and R.L.W. van de Weg, *A Distributed, Main-Memory Database Machine*, Proc. of 5th International Workshop on Database Machines, 1987.

[2] W. Bronnenberg, A. Nijman, E. Odijk, R. van Twist, *DOOM: A Decentralized Object-Oriented Machine*, IEEE Micro, October 1987.