# The CORAL Deductive Database System*

**Raghu Ramakrishnan**†
**William G. Roth**
**Praveen Seshadri**
**Divesh Srivastava**
**S. Sudarshan**‡

## 1 Introduction

CORAL [4, 5] is a deductive database system that supports a powerful declarative query language. The language supports general Horn clause logic programs, extended with SQL-style grouping, set-generation, and negation. Programs can be organized into independently optimized modules, and users can provide optimization hints in the form of high-level annotations. The system supports a wide variety of optimization techniques. There is an interface to C++ that enables programs to be written in a combination of imperative and declarative styles; C++ code can be called from declarative programs, and vice versa. A notable feature of the CORAL system is that it is extensible. In particular, new data types can be defined, and new relation and index implementations can be added. An interface to the EXODUS storage manager [2] provides support for disk-resident data, transactions and crash-recovery.

CORAL is available as source code (C++) from *ftp.cs.wisc.edu* at no charge. It is distributed with extensive documentation, including a tutorial user manual and a large suite of example programs. It is currently installed at over 125 sites and is being used for instruction as well as in research projects.

## 2 Goals of the Demonstration

The demonstration is intended to illustrate some of the important features of the system, to exhibit a wide range of programs that benefit from these features, and to emphasize that substantial applications can be and have been developed using CORAL. The demonstration will also display the range of evaluation techniques supported by CORAL, and give a feel for the performance of CORAL.

The features demonstrated include the following:

- general recursive rules
- non-stratified negation, aggregation and set-generation
- manipulation of set-values
- non-ground data structures such as difference-lists
- interface to C++, in particular, its use in supporting extensibility
- support for modules
- disk-resident data
- debugging support, including a trace facility and a graphical explanation package.

We also illustrate the following issues related to program evaluation:

- program transformations such as Magic Templates (and variants), Context Factoring and projection pushing.
- run-time techniques including Semi-Naive Evaluation (and variants) and Prolog-style "pipelined" evaluation.
- the use of high-level annotations to optionally guide optimization by providing hints on the choice of indexes, the use of duplicate checks, and the choice of rewriting and run-time methods, on a per-module basis.

*This research was supported by a David and Lucile Packard Foundation Fellowship in Science and engineering, a Presidential Young Investigator Award, with matching grants from Digital Equipment Corporation, Tandem and Xerox, and NSF grant IRI-9011563.

†The address of the first four authors is Computer Sciences Department, University of Wisconsin, Madison, WI 53706, U.S.A. These authors' email addresses are {raghu, roth, praveen, divesh}@cs.wisc.edu.

‡The address of this author is AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974, U.S.A., and his email address is sudarsha@research.att.com. This author's work was performed largely while he was at the University of Wisconsin, Madison, and was partially supported by the grants listed above.


SIGMOD /5/93/Washington, DC,USA
© 1993 ACM 0-89791-592-5/93/0005/0544...$1.50

With respect to performance, we would like to make the following points:

- good performance is achieved on a very wide range of programs, from simple joins and data-intensive recursive queries to Prolog-style programs.
- performance can often be tuned by a careful organization of the program in conjunction with some annotations.
- program compilation is fast, making interactive program development convenient.

# 3 Programs Demonstrated

We demonstrate a collection of programs that reflect the diversity of the CORAL system.

## Mimsy

Mimsy is a system that provides a user-friendly menu-driven interface for specifying sophisticated queries on stock market data. The queries are evaluated using CORAL, and the results can be viewed graphically as well as in textual form. Mimsy is inspired by the MIM [3] sequence analysis system. It provides much of MIM's functionality, and in addition, provides the important feature of being extensible. The development of the Mimsy package clearly demonstrates: (1) the importance of the CORAL/C++ interface and the extensibility of CORAL, (2) the advantages of having a powerful declarative language, and (3) the ability of the CORAL system to support substantial applications.

Examples of queries that can be posed using Mimsy include "Show the close of IBM on all days when it rose by 10%", "Show all increasing runs in IBM", "Show the close of IBM whenever IBM crosses DEC, and the weekly move of the Dow is over 10%". Mimsy uses data from the CRSP database distributed by the University of Chicago.

## Explain

The Explain system [1] is a graphical package for generating explanations of CORAL programs. In addition to providing a useful facility in the CORAL system, this package is of additional interest since its implementation uses CORAL extensively. The design is based on the representation of a program evaluation as a set of derivation trees. The system allows the user to view multiple derivations side-by-side and switch smoothly between the derivation of facts and the use of facts in other derivations.

## Transitive Closure Queries

We demonstrate several transitive-closure related queries, including single-source selections, full closure, shortest-path queries, currency arbitrage-style queries, and bill-of-material queries.

In addition to being excellent examples of programs suited to deductive databases, some of these programs illustrate advanced features such as non-stratified aggregation, the use of modules to structure programs, and the use of annotations to tune program performance.

## Dynamic Programming

These are programs that are again well-suited to deductive database style evaluation. The programs include the well-known knapsack and matrix multiplication examples.

## Non-Ground Tuples

We illustrate the advantages of supporting non-ground tuples using programs such as the Towers of Hanoi and natural-language parsing.

## C++ Interface

The examples include a program that embeds declarative code in C++ code, using extensions to C++ to directly access CORAL data, and a program that uses C++ to define predicates that can be used in declarative CORAL rules.

# References


[1] T. Arora, R. Ramakrishnan, W. G. Roth, P. Seshadri, and D. Srivastava. Explaining program evaluation in deductive systems. Submitted, 1993.

[2] M. Carey, D. DeWitt, G. Graefe, D. Haight, J. Richardson, D. Schuh, E. Shekita, and S. Vandenberg. The EXODUS extensible DBMS project: An overview. In *Readings in Object-Oriented Databases*. Morgan-Kaufman, 1990.

[3] Logical Information Machines. *The XMIM Reference Guide*, 2.1.1 edition, July 1992.

[4] R. Ramakrishnan, D. Srivastava, and S. Sudarshan. CORAL: Control, Relations and Logic. In *Proceedings of the International Conference on Very Large Databases*, 1992.

[5] R. Ramakrishnan, D. Srivastava, S. Sudarshan, and P. Seshadri. Implementation of the CORAL deductive database system. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1993.