Paradise: A Database System for GIS Applications*

The Paradise Team
Computer Sciences Department,
University of Wisconsin, Madison
paradise@cs.wisc.edu

1 Introduction

The goal of the Paradise project is to apply object–oriented and parallel database technology to the task of implementing a parallel GIS system capable of managing extremely large (multi-terabyte) data sets such as those that will be produced by the upcoming NASA EOSDIS project [Car92]. The project is focusing its resources on algorithms, processing, and storage techniques, and not on making new contributions to the data modeling, query language, or user interface domains.

At the outset, we organized the Paradise project as two phases. The goal of first phase was to produce a client–server version of Paradise. The second phase of the project is to add support for tertiary storage and extend the software to run on clusters of workstations and "shared nothing" multiprocessors [Sto86]. Phase one of the project is now complete and has produced a usable, client–server version of the system whose performance and functionality is comparable to other integrated GIS systems [DKL+94].

2 Data Model and Query Language

Paradise provides an extended–relational data model for modeling GIS applications and uses an extended version of SQL as its query language. The data model provides three type constructors: extent, tuple, and reference. An **extent** consists of a set of objects of the same type. A Paradise database consists of one or more such extents. Objects themselves are defined using the **tuple** type constructor. Each attribute can be an instance of either a standard base type (i.e. integer, float, string, ...), one of the predefined GIS–specific abstract data types (ADTs) including polygon, polyline, and raster, or a typed **reference** to another object.

3 Software Architecture

Version 1.0 of Paradise employs a query–shipping, client–server architecture. The front–end sends queries to the Paradise server for execution. After executing the query, the server ships the result objects back to the client process.

Client: The front–end provides a graphical user interface that supports querying, browsing, and updating of Paradise objects through either its graphical or textual interfaces.

*This work was partially supported by NASA Contracts #USRA–5555-17 and #NAGW–3895 and by an IBM Research Initiation Grant.

Server: The Paradise server uses SHORE [Shore] as its underlying persistent object manager. The Paradise server is implemented as a SHORE Value Added Server (VAS) directly on top of the SHORE Storage Manager. To the basic SHORE server, Paradise adds a catalog manager, an extent manager, a tuple manager, a query optimizer and execution engine, and support for point, polyline, polygon, and raster ADTs.

In designing and implementing the Paradise server, careful attention was paid to insure that the system could efficiently process queries (especially those involving spatial attributes) on large volumes of data. To meet this end, Paradise uses R*–trees [BKSS] (implemented with full concurrency control and recovery) as an access method. Bulk loading of data into an R*–tree and clustered indices on spatial attributes are supported. To efficiently handle raster data, the implementation of the raster ADT in Paradise breaks large raster images into smaller chunks and uses compression for storing the chunks in the database. This technique speeds up the processing of queries involving raster data [DKL+94].

4 Contents of The Paradise Demo

The demonstration will illustrate the spatial data handling features of the Paradise server and the GUI capabilities of the Paradise client. The first part of the demonstration will exhibit the query processing features using land map data for Europe. This part will include demonstration of features that highlight the vector (polygons, polylines, points ...) based querying functionality of Paradise. The second part of the demo will use data from the Sequoia 2000 storage benchmark [SFGM] to demonstrate the raster handling capabilities of Paradise. The final part of the demo will illustrate the video handling capabilities of Paradise.

References

[BKSS] N. Beckmann, Hans-Peter Kriegel, R. Schneider, and B. Seeger. "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles". In *Proc. 1990 SIGMOD Conf.*

[Car92] R. V. Carlone. "NASA's EOSDIS Development Approach". Technical report, United States General Accounting Office, February 1992.

[Shore] M. J. Carey et at. "Shoring up Persistent Objects". In *Proc. 1994 SIGMOD Conf.*

[DKL+94] D. J. DeWitt, N. Kabra, J. Luo, J. M. Patel, and J. Yu. "Client–Server Paradise". In *The 20th VLDB Conf.*, Santiago, Chile, September 1994.

[SFGM] M. Stonebraker, J. Frew, K. Gardels, and J. Meredith. "The SEQUOIA 2000 Storage Benchmark". In *Proc. 1993 SIGMOD Conf.*

[Sto86] M. Stonebraker. "The Case for Shared Nothing". *Database Engineering*, 9(1), 1986.