Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

An Object-Oriented Approach to Astronomical Databases

R. J. Brunner, K. Ramaiyer[1], A. Szalay, A. J. Connolly

Department of Physics & Astronomy, The Johns Hopkins University, Baltimore, MD 21218

R. H. Lupton

Department of Astronomy, Princeton University, Princeton, NJ 08544

Abstract. The rate at which we can acquire astronomical data outstrips our ability to manipulate and interpret it using standard techniques. Future digital astronomical surveys will enter the terabyte regime. In addition, the actual data themselves will be complex, comprising images and spectra, as well as the traditional measured attributes. We describe here an implementation of a prototype object-oriented database, distributed in a client-server framework that is capable of both handling these volumes of data and facilitating geometric queries in an efficient manner.

1. Introduction

With the size of astronomical databases increasing into the terabytes, new approaches must be taken to store, manage, and access these datasets. As an example, consider the Sloan Digital Sky Survey (Gunn & Knapp 1992), a multicolor digital survey of the north Galactic cap which will produce images in 5 bands of more than one hundred million objects, as well as a spectroscopic survey that will obtain more than a million spectra. A single archive site for the entire set of data produced by this survey will require tens of terabytes in permanent storage media. Under the current approach to astronomical databases, statistical studies will be hampered by both long query times and the enormous amounts of data that must be examined.

Our proposed solution is to store the data in an object-oriented database using a multidimensional indexing scheme: the $k-d$ tree. The data will be distributed in a client–server architecture to provide the maximum flexibility as well as retain portability. The user will formulate queries which are then transformed into data requests and routed to the appropriate databases where the requested data are extracted and returned. Section 2 discusses the general design principles behind our system. Section 3 describes the data organizational model we employ, and in particular the $k-d$ tree. Finally, section 4 outlines the data flow model and our prototype implementations.

---

[1]Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218

2. Design Principles

Our data archiving scheme is based on two fundamental principles: object-oriented programming (OOP), and distributed processing via a client–server architecture. Before describing the technical aspects of our system, a brief digression into the reasoning behind the choice of these principles should prove useful.

2.1. The Object-Oriented Philosophy

Historically, astronomical databases have been stored in a relational format. The previously 'flat' data was easily adapted to the tabular format of relational records. These databases would then be indexed by a limited number of key fields, providing rapid responses to queries involving indexed fields. The main advantages of this approach are the stability of the commercial products and the standard query language (SQL) available for interacting with the database.

The object-oriented philosophy is a fundamental shift away from the relational paradigm. Instead of following the traditional path of formulating an algorithm, and then forcing the data to work within the program, OOP utilizes the fundamental relationships within the data to naturally implement the appropriate algorithm. Object-oriented databases extend this philosophy, modeling the database on a schema (parameter lists). Thus, the database becomes a distribution of objects with associated properties instead of a table of values. The objects within the database are then compared, stored, and operated on as individual units instead of their fundamental parameters.

All object-oriented database management systems that follow the Object Database Standard (ODMG–93) must permit a C++ interface, providing efficient access to the database. As a result, these databases are extremely fast in extracting large amounts of information often approaching the inherent I/O bandwidth limitations.

Another ability of OOP is the ability to persistently model complex data structures. As an example, the schema for an object from an astronomical survey might consist of Galactic coordinates, multicolor fluxes, deblended multicolor images, associated spectra, related housekeeping information, and even references to objects in external databases. This object would then be manipulated as a fundamental unit in the same fashion as the base datatypes: char, integer, and float. The attributes of an object are extracted by methods, allowing for a dynamic update of the data without rebuilding the entire database. Therefore, the implementation of recalibrations of the astrometry or photometry is accomplished by merely modifying the appropriate access functions.

2.2. Client–Server Architecture

A client–server architecture serves to distribute the processing load over multiple computers in a redundant fashion. This results in a more flexible system which can be optimized to avoid congested archive sites in processing queries. Additionally, each process is isolated, allowing commercial products to be integrated into the system with minimal impact and providing cross platform connectivity. Such a system is easily expandable to more client/server processes or databases without disturbing the rest of the archival system.

Using multiple archive servers can extend the data retrieval into a parallel I/O operation, increasing the access efficiency. With the future development of massively parallel I/O machines, an inherently client–server architecture will be quickly adapted. Essentially, this approach hides the details of the processing from the user, providing only the results.

3. Data Organizational Model

The $k-d$ tree (Friedman, Bentley, & Finkel 1977; Samet 1990) is a multidimensional indexing scheme in which d dimensional data is indexed in k dimensions. Each tree node represents a subvolume of the parameter space, with the root node containing the entire k dimensional volume spanned by the data. A balanced binary tree is constructed by splitting each node into two sections along the median of the k dimension which will maximize the clustering of data in the two child nodes. The k dimensions can be actual parameters or the principal components of the parameter data.

The number of levels in the tree is determined by maximizing the number of objects in a leaf node (container) with the constraint that the $k-d$ tree must be small enough to reside in the memory of the user's computer. All objects in a leaf node are then stored together on the physical media. As a result, friends are inherently stored together, producing a built-in index.

Since each node knows its actual boundaries in the k dimensions, the $k-d$ tree can also serve as a coarse grained density map of the actual data parameters. Queries are first performed on the $k-d$ tree, limiting the query volume to those leaf nodes that fall within the query, prior to the entire database, resulting in predictions for the search time, and estimated number of objects satisfying the search. This provides a feedback feature for the user, which can minimize costly queries.

Database queries can be modeled as cuts within the parameter space of the database. These parameter cuts can be combined using boolean operators to create complicated query volumes. Geometric queries can be constructed through linear combinations of the parameters, proximity cuts, and k-nearest neighbor searches. These geometric queries utilize the nodal boundaries to determine the containers that intersect the query volume and must be searched in more detail. Using multiple linear combinatorial cuts, one can create convex hulls (polyhedra) that carve out a subspace of the parameter volume. This allows associative queries to be performed, by first constructing a convex hull from a given sample of objects, and then finding all other objects in the database that lie within the given convex hull.

4. Data Flow Model

The data flow model for our system involves four main components: the object-oriented database management system (OODBMS), the object request broker (ORB), the client process, and the graphical user interface (GUI).

Our prototype system currently employs Objectivity, a commercial object-oriented database management system, for maintaining the persistence of our objects. The $k-d$ tree and the actual data objects are retrieved from the

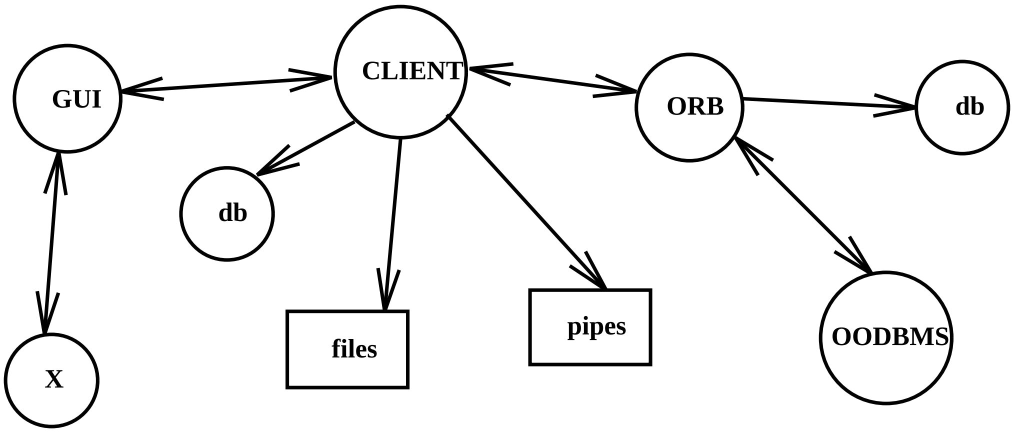

Figure 1. The data flow model.

OODBMS using an object request broker (ORB) which is the only process in our system that is syntactically tied to the OODBMS. The ORB becomes a server, awaiting TCP/IP socket connections from client processes. Together, the OODBMS and the ORB are the only process that physically reside, along with the actual data, at the archive site.

The client process is responsible for interpreting the queries using the $k-d$ tree to determine containers that satisfy the search, from which an estimate for the returned number of objects and search time are determined. The client also must establish the connection to the ORB, initially retrieving first the $k-d$ tree and then the desired objects. The extracted data can be piped directly into an analysis tool, saved into a file, or made locally persistent. The client interacts with the user via the GUI, which facilitates query construction and data retrieval.

We have adopted this model in two implementations. The first prototype involved an all sky simulation containing two million objects with eight parameters: right ascension (RA), declination (DEC), five colors (U, G, R, I, Z), and redshift. The $k-d$ tree was constructed with ten levels involving splits in the order: RA, RA, RA, DEC, DEC, DEC, G, G, R, and R. Complex queries involving multiple parameters are one hundred times faster than simple linear searches. Currently, only boolean combinations of parameter cuts and linear combinations of parameters are supported. A second prototype using the *IRAS* point source catalog is under construction.

References

Friedman, J. H., Bentley, J. L., & Finkel, R. A. 1977, ACM Trans. Math. Software, 3, 209

Gunn, J. E., & Knapp, G. R. 1992, PASP, 43, 267

Samet, H. 1990, The Design and Analysis of Spatial Data Structures (Reading, Addison-Wesley)

The Object Database Standard: ODMG-93, ed. R. Cattel (New York, Morgan Kaufmann)