**REPORT DOCUMENTATION PAGE**

**1. AGENCY USE ONLY (Leave blank)**

**2. REPORT DATE**
1995

**3. REPORT TYPE AND DATES COVERED**
Technical Report

**4. TITLE AND SUBTITLE**

Storage and Visualization of Spatial Data in a High-Performance Semantic Database System.

**5. FUNDING NUMBERS**

DA04-94-6-0024

**6. AUTHOR(S)**

N. Rishe, D. Barton, M. Sanchez

**7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)**

Florida International University  
School of Computer Science  
University Park Campus  
Miami, FL 33199

**8. PERFORMING ORGANIZATION REPORT NUMBER**

FIU TR # 95-15

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

U.S. Army Research Office  
P.O. Box 12211  
Research Triangle Park, NC 27709-2211

**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

ARO 32-427.4-MA-SXI

**11. SUPPLEMENTARY NOTES**

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

**12a. DISTRIBUTION / AVAILABILITY STATEMENT**

Approved for public release; distribution unlimited.

**13. ABSTRACT (Maximum 200 words)**

We are developing a prototype massively parallel database management system with applications to earth sciences. Our system will enable the highly efficient accumulation and retrieval of vast amount of general, scientific, and spatial data, utilizing a semantic/object-oriented approach to database management. One type of data in this system is a generalized spatial function - a function from a Cartesian product of several continuous and/or discrete domains into a Cartesian product of continuous domains and/or discrete domains and/or sets of semantic facts. This paper addresses issues of database storage of such functions, their querying, and visual presentation of results as multi-dimensional objects, particularly superimposition of two spatial functions in a 3-D display.

**14. SUBJECT TERMS**

semantic databases, spatial data, high performance, massive parallelism, scientific data, earth sciences databases, object-oriented databases, spatial queries.

**15. NUMBER OF PAGES**

340-01-280-5500

**16. PRICE CODE**

UL

**17. SECURITY CLASSIFICATION OR REPORT**

UNCLASSIFIED

**18. SECURITY CLASSIFICATION OF THIS PAGE**

UNCLASSIFIED

**19. SECURITY CLASSIFICATION OF ABSTRACT**

UNCLASSIFIED

**20. LIMITATION OF ABSTRACT**

UL
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.
Storage and Visualization of Spatial Data in a High-performance Semantic Database System

by

N. Rishe, D. Barton, and M. Sanchez

Technical Report #95-15
Storage and Visualization of Spatial Data in a High-performance Semantic Database System
FIU SCS Technical Report 95-15

Naphtali Rishe, David Barton, and Mario Sanchez

High-performance Database Research Center
School of Computer Science
Florida International University
University Park, Miami, FL 33199
(305) 348-2025, rishen@fiu.edu

We are developing a prototype massively parallel database management system with applications to earth sciences. Our system will enable the highly efficient accumulation and retrieval of vast amount of general, scientific, and spatial data, utilizing a semantic/object-oriented approach to database management. One type of data in this system is a generalized spatial function — a function from a Cartesian product of several continuous and/or discrete domains into a Cartesian product of continuous domains and/or discrete domains and/or sets of semantic facts. This paper addresses issues of database storage of such functions, their querying, and visual presentation of results as multi-dimensional objects, particularly superimposition of two spatial functions in a 3-D display.

Keywords: semantic databases, spatial data, high performance, massive parallelism, scientific data, earth sciences databases, object-oriented databases, spatial queries, visualization, generalized spatial functions, spatial function superimposition.

1. INTRODUCTION

Earth science database applications have three essential needs:

- strong semantics embedded in the database so as to effectively handle the complexity of information
- storage of spatial, image, and other non-conventional data
- very high performance facilitating massive data flow

This research was supported in part by NASA (under grant NAGW-4080), ARO (under BMDO grant DAAH04-0024), NATO (under grant HTECH.LG-931449), NSF (under grant CDA-9313624 for CATE Lab), National Park Service (CA-5280-4-9044 and CA5280-0-9018) and State of Florida.
Abundant evidence demonstrates that semantic/object-oriented databases can better satisfy the first two needs than relational databases. We are developing a highly parallel database machine based on the semantic/object-oriented approach that will also satisfy the third need — high performance.

Our research aims to significantly improve the usability and efficiency of highly parallel database computers and machine clusters (tightly networked groups of machines). Our prototype database management system will have substantial advantages over current database machines, due to:

- **Usability.** Our object-oriented system is based on the Semantic Binary Model of databases, unlike most current database systems, which are mainly based on the Relational Model. The use of semantic models ensures better logical properties: friendlier and more intelligent generic user interfaces based on the stored meaning of the data, comprehensive enforcement of integrity constraints, greater flexibility, and substantially shorter application programs.

Semantic databases represent information as a collection of objects and relationships between these objects. The Semantic Binary Model of databases is a semantic model with object-oriented features [Rishe-92-DDS]. Data items related to objects can be of arbitrary size, multi-valued, or missing entirely. We have applied this approach to various types of data, including scientific and multi-media data. Semantic objects are not required to be identified by keys. An object may belong to many categories at the same time. Inclusion of categories determines inheritance of properties.

- **Efficiency.** Our system will be more efficient than existing database machines. This higher-efficiency goal can be attained by exploiting the system’s understanding of the data’s semantics and due to the higher abstraction level. The algorithms and prototype system that we are developing are highly efficient for both small and massive numbers of processors equipped with separate memories and storage devices. In particular, the use of the semantic model allows better exploitation of parallelism, by providing a means of distributing data among these processors in a way which is invisible to both database programmers and database users. We are developing algorithms and prototype software for the outer levels of the system (intelligent query processors and optimizers, content accessibility), as well as inner-level storage management. These algorithms will then be combined into one high performance system, with a very efficient representation of tempo-spatial and fuzzy data. Data is stored in highly-compressed form while allowing efficient and flexible retrieval.

This paper presents our theory and algorithms associated with one of the data types in our system: a *generalized spatial function* — a function from a Cartesian product of several continuous and/or discrete domains into a Cartesian product of continuous domains and/or discrete domains and/or sets of semantic facts. For example, ocean temperature is a function $f: X \times Y \times Z \times T \times O \rightarrow R^2 \times \text{Factsets}$, where $X \times Y \times Z \times T$ is the space-time continuum and $O$ is a discrete set of observation stations that reported measurements. Thus, $f(x, y, z, t, o) = (s, i)$, where $s$ is a segment of temperatures (e.g., 50 degrees plus or minus 0.01 degrees) and $i$ is a set of semantic facts. Another example is remote sensing (photography) of ocean color by the SeaWiFS satellite.
The spectrum of problems we have addressed concerning this data type includes:

1. Highly-efficient basic queries, including "inverse" queries (e.g., "Where is the temperature of about 70 degrees?")
2. Compact lossless storage
3. Compact lossy storage, particularly by approximating function values.
4. Efficient complex queries
5. Load balancing between processors and storage units
6. Visual presentation of query results as animated movies.
7. Visual presentation of query results containing two spatial functions of the same space as a 3-D overlay. For example, ozone layer thickness represented as elevation is superimposed with temperature represented as color.

2. RELATED WORK

Spatial and scientific databases have attracted the attention of many researchers. The proper statistical analysis of spatial and spatiotemporal data is critical to the success of any scientific study which uses such data. Cressie [Cressie-91] provides an extensive coverage of the current theories and methods used for spatial analysis, with some discussion of spatiotemporal methods. Our system will support these types of analyses, although the statistical procedures themselves are not part of our project.

A long-term goal of the JGOFS project [Flierl&al.-93] is to establish strategies for observing changes in ocean biogeochemical cycles in relation to climate change. A distributed approach is used in that project, where the data are not gathered into a central archive but rather reside at the originator's site. An object-oriented database system is developed for this purpose.

It is necessary to develop management tools that offer both the functionality required by a scientific environment and an interface that feels natural and intuitive to the non-expert [Ioannidis&al.-93]. A desktop Experiment Management System has been proposed as the interface between the experimental scientists and the data [Ioannidis&Livny-92].

The use of "layered database technology" to support scientific applications is suggested in [Shoshani-93]. It allows to provide interfaces to various levels for different scientific applications.

Research has been done on object-oriented data management systems for physical scientists [Hachem&al.-92]. The current goal of the Gaea project is to construct a prototype which permits integration of heterogeneous and complex datatypes in geography [Hachem&al.-93].

One of the ways to provide database support for high performance scientific applications is to create a special language for describing, finding, and accessing data by applications programs [Pfaltz&French-93]. The main goal is to interface many different computing environments to a common, persistent data space [Pfaltz&al.-88].
[Smith & al. – 93] considers models using a large and heterogeneous collection of datasets. The problem of coupling several complex models arises in the application of spatially-distributed models of water, sediment and solute transport in the Amazon basin. A high-level Modeling and Database Language (MDBL) is suggested.

The need of using data from different sources arises in scientific applications. For example, the use of object-oriented databases for this purpose in the domain of computational chemistry is discussed in [Cushing & al. – 92], and [Cushing & al. – 93].

The SEQUOIA 2000 project ([Stonebraker – 93], [Stonebraker & al. – 93]) is designed for global change research. Management, storage and access to massive amount of data, are considered in that project.

Some interesting problems appear in medical applications of database technology. One of them is management of large repositories of image, text, and scientific information generated by academic medical research centers. Another one is the integration of different independent database systems which already exist in many specialized branches of medicine. These problems as well as the extension of traditional object-oriented data models into the temporal domain for accurately representing the data stored in medical image databases are considered in [Cardenas & al. – 93] and [Chu & al. – 92]. Visualization methods for query results and handling of 3-dimensional spatial data sets created from 2-dimensional medical images are investigated in the QBISM project [Arya & al. – 93].

3. GENERALIZED SPATIAL FUNCTIONS

This section defines a new data type: a generalized spatial function.

Consider spatial functions that map a Euclidean space into values: \( f : R^n \rightarrow R^m \) (where \( R \) is the continuum of real numbers). For example, ocean temperature is an \( R^4 \rightarrow R \) function of latitude, longitude, depth, and time.

In some spatial applications, the function's domain may include discrete dimensions. For example, if Observers is a discrete set of observing devices then the perceived ocean temperature is \( R^4 \times \text{Observers} \rightarrow R \).

In some spatial applications, each point in space is assigned not only certain values but also other arbitrary information, which can be generalized as a set of facts. Let Factsets be the set of all finite sets of facts. Let D be a discrete domain. A generalized spatial function:

\[
 f : R^m \times D^n \rightarrow R^k \times D^p \times \text{Factsets}^j
\]

where \( m > 0, n >= 0, k >= 0, p >= 0, \) are integers, and \( j \) is 0 or 1.

For example, if facts can be observed as associated with certain space-time regions then the ocean temperature is: \( f : X \times Y \times Z \times \text{Time} \times \text{Observers} \rightarrow \text{Temperature} \times \text{Factsets} \).

In observation of natural phenomena one needs to distinguish between raw and processed/interpreted views. A raw view is a discrete set of measurements as reported by measurement devices. Raw views contain noise. Further, raw views are typically expressed in terms of the activity of the measuring devices rather than in terms of the actual coordinates of the observed phenomena, i.e. the coordinates are not geo-calibrated. Processed/interpreted views attempt to estimate and approximate the actual attributes of the observed phenomena, by what we call generalized spatial functions. The broad community of users of spatial data
needs only processed views. A small group of scientists uses the raw views to derive processed views by applying various theories. These scientists keep improving and fine-tuning such theories. Consider, for example, the problem of translating a satellite's motion coordinates and device angles into the Earth surface coordinates of the measured phenomenon. The techniques of doing so, and their precision, are open for improvement, especially in the polar zones. The raw spatial data cannot be smoothened and must be available in full detail. The processed spatial data represents an estimation of the natural phenomenon, and, therefore, has properties of typically continuous functions defined over the space continuum. The raw spatial data may coexist in a database with processed data, but different techniques may be used for storage and retrieval of the two types.

As an example, consider the ocean color observations to be made by the NASA SeaWiFS satellite. Figure 3-1 represents the semantic schema that we use in our experimental database to store and access raw SeaWiFS data. Actually, this not the immediate satellite data but rather derived by a straightforward algorithm, not involving the intelligence and ambiguity of estimating the actual color in ocean-surface coordinates.

![Figure 3-1](image)

**Figure 3-1.** SeaWiFS: Logical Schema of Unpacked Raw Data. Only some pixels directly correspond to some status records; for each such status record only some of the data from the list is available.

The schema of Figure 3-1 uses the semantic modeling notation of [Rishe-92-DDS], which is briefly explained here. The category PIXEL is the set of pixels, in orbital coordinates, for which color recording is taken. The attributes Scan_coordinate and Orbit_propagation_coordinate are of type Integer and define a matrix of pixels. For each pixel, there are 8 color values for 8 different bandwidths (they are taken by eight devices on board). This vector of eight values is represented by the attribute Sensed-values of type $\mathbb{R}^8$. The category PIXEL contains the actual measurements as taken. What geographical points on the ocean surface does a given pixel correspond to, and what is the real color of those points, is an issue of non-trivial and imprecise analysis. Such analysis is aided by the other information in this schema. Some pixels are associated with satellite status data by the relation BY. This relation is many-to-many. It is not a total relation, meaning that some (actually, most) pixels do not have any satellite status data explicitly associated with them.
(but such data can be inferred from that associated with nearby pixels; we say that a status record is explicitly associated with a pixel if the record was transmitted by the satellite immediately before the pixel). Since there is a variety of satellite status records, transmitted at various times, each status record may contain some, but not all, of the following information. The Position and Velocity attributes are vectors of three numbers each, representing the satellite’s position and velocity. Angle and Tilt refer to the aiming of the lens. Their type is Number, referring to any number representable by a finite string of digits. We do not have any precision or magnitude restrictions on real numbers [Rishe-92-IB]. Telemetry_data is a multi-valued attribute of type String, i.e. one satellite status record can have a set of strings of telemetry data. Mode is an attribute of an enumerated type. Time is an attribute of the type Date-time, implemented as arbitrary numbers, standing for the number of seconds since a certain date-time \( t_0 \). (It allows any decimally-expressible fraction of a second.)

Figure 3-2 is the semantic schema of processed SeaWiFS data, further interpreted to expand the discrete measures to be reflective of the space-time continuum that they represent. It defines generalized spatial function over the three-dimensional continuum of ocean surface latitude longitude and time. Most users are interested only in this interpreted information, not in the raw data. The following section discusses an efficient implementation of this "infinite" data structure.

<table>
<thead>
<tr>
<th>POINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>latitude: Number</td>
</tr>
<tr>
<td>longitude: Number</td>
</tr>
<tr>
<td>at: Date-time</td>
</tr>
<tr>
<td>Values: R*8</td>
</tr>
</tbody>
</table>

Figure 3-2. SeaWiFS: Logical schema of interpreted data. There is an infinite continuum of points.

4. PHYSICAL STORAGE OF GENERALIZED SPATIAL FUNCTIONS

4.1. Goals

Logically, spatial functions are defined over a continuum. At the physical level, we represent spatial functions by a finite set allowing approximate interpolation of the function. We notice that the spatial function itself represents an estimation of a natural phenomenon, derived from some finite raw data. Therefore the values interpolated from its physical representation need not be more precise than said estimation.

As an example of a characteristic problem and its solution, let us consider ocean temperature. The ocean can be regarded as a four-dimensional Euclidean space of longitude, latitude, depth, and time. Thus, temperature \( T \) is a function \( T(x,y,z,t) \). Additionally, there is a discrete dimension of observation sources, which may disagree between them. Thus, the temperature function may have five arguments: \( T(x,y,z,t,s) \). If \( \delta \) is the precision of
knowledge of temperature at a point, then the assertion of the database is that the actual temperature is between \( T(x, y, z, t, s) - \delta \) and \( T(x, y, z, t, s) + \delta \). In some application, \( \delta \) is not a constant but depends on the point, in which case we have generalized spatial functions producing for each point a segment of possible temperature values: \( T(x, y, z, t, s) \pm \delta(x, y, z, t, s) \). If the database represents this temperature knowledge fully, we call it a **fully lossless** representation of the interpreted spatial data. If this knowledge is approximated in the database by a value segment \( T'(x, y, z, t, s) \pm \Delta \) containing the segment \( T \pm \delta \) and \( \Delta \) is not substantially greater than \( \delta \), then we call this representation **approximately lossless**. In this case, the difference \( \Delta - \delta \) is the degree of approximation. As will be discussed below, we can vary the degree of approximation as a function of the required compactness and efficiency of the database. If for some points \( (x, y, z, t, s) \), \( T'(x, y, z, t, s) \) is outside the segment \( T(x, y, z, t, s) \pm \delta \), then the representation is **lossy**. When the generalized spatial function is continuous (and it typically is for interpreted natural phenomena, except for boundary conditions) we can have a highly compact and efficient fully- or approximately lossless representation, as will be shown later in this paper.

The following are examples of some queries that are asked about this function:

\((Q_1)\) Find the temperature for a given 5-dimensional point (whether it is a point of actual measurement or interpolated). This is the most basic query.

\((Q_2)\) a more complex query: find the temperature of a four-dimensional space-time point independent of the observation source, obtained by weighing the different sources according to their known reliability, etc.

\((Q_3)\) find the average temperature of a given arbitrary segment of space-time body.

\((Q_4)\) delineate space-time ranges where the temperatures are between given \( t_1 \) and \( t_2 \).

Logically, we assume that we have a virtual infinite database containing all the observed measurements and the interpolated points. The following specifications delineate this virtual database's schema, graphically depicted in Figure 4-1.

- **OBSERVATION-SOURCE** — category (The set of entities measuring ocean temperature)
  - **description** — attribute of **OBSERVATION-SOURCE**, range: String \(1:1\) (The identifying description of an observation source)
  - **reliability** — attribute of **OBSERVATION-SOURCE**, range: \(0..1.00\) \((m:1)\) (A number between 0 and 1)
- **POINT-VALUE** — category (The infinite set of all pairs of space-time points and their possible temperature values)
  - **point** — attribute of **POINT-VALUE**, range: \(R^4\) \((m:1)\) (Vector in space-time)
  - **temperature** — attribute of **POINT-VALUE**, range: Number \((m:1)\) (In degrees Kelvin)
- **produced** — relation from **OBSERVATION-SOURCE** to **POINT-VALUE** \((m:m)\) (A many-to-many association between point-values and observation sources that collected raw data which upon processing and interpretation yielded the point-value)
4.2. Linear Hyperquadrant Data Structure

The infinite logical view of Figure 4-1 can be mapped into a compact actual database of Figure 4-2. Later in this paper we will introduce further refinements of compactness and efficiency of this database.

In the database schema of Figure 4-2 we represent the space-time continuum as a hexadecimal tree of hyper-quadrants. We utilize the well-known theory of linear quad-trees of [Gargantini-82], which we extend to multi-dimensional generalized spatial functions and adapt to semantic databases. Our further refinements of this data structure are discussed in the next section.

Let $\delta$ be the precision of knowledge of the generalized spatial function and $\Delta$ be the desired precision of knowledge representation in the database, $\Delta \geq \delta$. In order to simplify this discussion, we will use the terms of the above temperature examples (whereby extending the results to an arbitrary generalized spatial function with an arbitrary number of dimensions will be obvious). Further, we will assume that $\Delta$ and $\delta$ are constants over the observed space-time. Generalization to the case when they are varying functions over the space-time is easy.

First we define the partitioning of a four-dimensional continuum into a tree of hyper-quadrants labeled by hexadecimal strings. The relevant space-time is bounded and, therefore, it can be embedded in a huge hyper-rectangle, $S$. Now, let us halve all the edges of $S$, thus partitioning $S$ into 16 hyper-quadrants touching each other at the center of gravity of $S$. Let us label them by the 16 hexadecimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f. Each of them can in turn be partitioned into 16 smaller hyper-quadrants, denoted by two hexadecimal digits, e.g. #7 is partitioned into 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 7a, 7b, 7c, 7d, 7e, 7f. Each of them can be further partitioned, and so on. The inclusion between hyperquadrants is defined by their label, so no pointers would be necessary: hyperquadrant $h_1$ contains hyperquadrant $h_2$ if and only if label($h_1$) is a prefix of label($h_2$).

We note that a mathematical point $(x,y,z,t)$ in space-time is a hyperquadrant of zero size; its label is an infinite hexadecimal string microhyperquadrant $(x,y,z,t)$. (We are introducing this only for the purpose of analysis below, not for actual storage in the database).

Now, a four-dimensional temperature function of space-time $T:R^4 \rightarrow R$ can represented, up to the desired degree of precision $\Delta$, as a finite set of non-overlapping hyper-quadrants of various sizes by the following recursive process: if the function varies on a given hyperquadrant $h_1$ more than allowed by the desired precision $\Delta$, then partition $h_1$ into sixteen smaller hyperquadrants.
Figure 4-2. Finite representation of a generalized spatial function
Temperature by a tree of hyper-quadrants

Now, the query \( Q_1 \) becomes:
\[
T(x,y,z,t,s) = \text{get m.TEMPERATURE where (s PRODUCED m and m.LABEL is a prefix of \( \text{microhyperquadrant}(x,y,z,t) \))}
\]

To allow efficient computation of queries like \( Q_1 \), the hyperquadrants of a given source \( s_0 \) can be stored at the physical level as a subfile HYPERQUADRANTS[label,temperature]. This subfile is a B+-tree ordered by labels. The following explains why \( Q_1 \) can be resolved in just one access to the disk.

The index level of the B+-tree contains the first labels of each physical block of records. Therefore, the index level is several orders of magnitude smaller than the data level. For example, if 1000 records [label,temperature] fit in each block, then the index level is 1000 times smaller than the data level of the B+-tree. Thus, we can normally assume that the index level resides in the memory (if this assumption were invalid, then the number of disk accesses to perform \( Q_1 \) would go from one to just two accesses). Since for \( s_0 \) the space-time was partitioned into a set of disjoint, varied-sizes hyperquadrants, there is only one hyperquadrant whose label \( l_1 \) is a prefix of \( \text{microhyperquadrant}(x,y,z,t) \). The label \( l_1 \) is thus the lexicographically greatest stored label less than \( \text{microhyperquadrant}(x,y,z,t) \). This record must reside in the data block whose first label \( l_0 \) is the lexicographically greatest index-level label below \( \text{microhyperquadrant}(x,y,z,t) \). Thus the index level will point to exactly one block containing the answer to query \( Q_1 \). Since one cannot possibly resolve a query requiring information from a disk in less than one disk access, the above-described algorithm is optimal.

This data structure also allows highly efficient computation of queries \( Q_2 \) and \( Q_3 \). To efficiently resolve query \( Q_4 \), which delineates areas having a temperature in a given range \( t_1 \) to \( t_2 \), we store an inverse index subfile INVERSE[temperature,label] which is a B+-tree ordered by temperature. The answer to \( Q_2 \) is the set of records \( \text{INVERSE}[t,l] \) where \( t_1 - \Delta \leq t \leq t_2 + \Delta \). This is the contiguous fragment of the file INVERSE specified as a B-tree range from \( t_1 - \Delta \) to \( t_2 + \Delta \). If the number of records in the output is substantially less than can fit into one data block then the query can normally be resolved in just one disk access. If the number of records in the output is large enough to fill \( n \) blocks then the query can normally be resolved in \( n+1 \) disk accesses. This is either the optimum or very close to the optimum.

We should note that Temperature, like all the numbers in our Semantic DBMS is of arbitrary precision and magnitude, and must be represented by a compact bit string of varying length. Further, in order to allow B-tree indexing and certain other operations, these varying-length strings must be lexicographically orderable preserving the meaningful order of numbers. To accomplish this, we use the order-preserving varying-length compact number
encoding defined in [Rishe-92-IB].

All the subfiles of the database are placed in one database file. In the high-performance version of our system, this file is partitioned between many disks and processors.

4.3. Polynomial Approximation Data Structure

Still further reduction in the storage of a generalized spatial function, e.g. the Temperature function, can be obtained if we sample the space-time not into very small bodies of approximately constant temperature (i.e. varying within a given $\Delta$ only) but into larger bodies whose temperature can be represented by an analytical function. For such bodies we will store the average temperature as well as an optional polynomial describing the offset in terms of the points' coordinates. This can also achieve a fully lossless representation without a great storage overhead.

Consider a generalized spatial function $T(x,y,z,t)\pm\delta(x,y,z,t)$, representing interpreted (processed) knowledge of a natural phenomenon, e.g. the ocean temperature. Since our knowledge of Nature is never exact, we can normally assume that $\delta>0$. We produce a fully lossless representation of this function by a finite set of non-overlapping hyper-quadrants of various sizes by the following recursive process:

Let $h$ be a hyperquadrant; let $\text{center}(h)$ be its center point; let $\text{average}(h)$ be the average temperature of $h$, defined as:

$$\text{average}(h) = \frac{\int_{(x,y,z,t)\in h} T(x,y,z,t) \, dx \, dy \, dz \, dt}{\text{volume}(h)}$$

Let $P_h : R^4 \to R$ be a minimal-degree polynomial function of the four-dimensional space such that:

(*) $\forall (x,y,z,t) \in h: |(\text{average}(h) + P_h((x,y,z,t) - \text{center}(h)) - T(x,y,z,t)| \leq \delta(x,y,z,t)$

For a polynomial $P$, let $\text{length}(P)$ be the sum of lengths of representations of the coefficients of the polynomial $P$. For example,

$$\text{length}(<x^2+1.35178>) = \text{length}(2)+\text{length}(1.35178)=1+6=7.$$  

Let $\text{Maxlength} \leq \infty$ be a global constant limiting the length of polynomials we wish to store in the database.

Recursive procedure on hyperquadrant $h$: If a polynomial $P_h$ satisfying (*) does not exist, or cannot be found efficiently, or if the length of minimal such polynomial exceeds $\text{Maxlength}$, then partition the hyperquadrant $h$ into 16 contained hyperquadrants and recursively apply the procedure to each of the 16 hyperquadrants.

To further reduce the database's size, we combine hyperquadrants into clusters if they can use the same polynomial. A cluster is one hyperquadrant or a set $C$ of hyperquadrants (typically adjacent but not necessarily of the same size) with a polynomial $P_C$ such that:

$$\forall h \in C: \forall (x,y,z,t) \in h: |(\text{average}(C) + P_C((x,y,z,t) - \text{center}(C)) - T(x,y,z,t)| \leq \delta(x,y,z,t)$$

where $\text{center}(C)=\frac{\sum_{h \in C} \text{center}(h)}{\text{cardinality}(C)}$,  

$$\text{average}(C)=\frac{\sum_{h \in C} \text{average}(h) \times \text{volume}(h)}{\sum_{h \in C} \text{volume}(h)}$$
Figure 4-3 defines a schema of the resulting compact and efficient database.

<table>
<thead>
<tr>
<th>OBSERVATION SOURCE</th>
<th>produced</th>
<th>CLUSTER</th>
</tr>
</thead>
</table>
| description: String 1:1 | (many-to-many) | hyper-quadrant: HexadecimalString  
average-temperature: Number dgr-Kelvin  
offset-function: Polynomial |
| reliability: 0..1.00 |                |                                                                         |

**Figure 4-3.** One cluster has several hyper-quadrants and an optional offset interpolating function

Further, to avoid storing the polynomial in every cluster, we can have a global default linear polynomial, which will be implied for those clusters for which no polynomial has been assigned in the databases. Now, Query $Q_1$ becomes:

$$T(x, y, z, t, s) =$$

get m.AVERAGE-TEMPERATURE+ (m.OFFSET-FUNCTION)((x,y,z,t) – center(m))

where s PRODUCED m and m has a hyper-quadrant h which is a prefix of the hexadecimal string microhyperquadrant (x,y,z,t).

In our database implementation, the above query will normally be resolved in just two disk accesses, assuming there is only one observation source covering the point (x,y,z,t).

Queries $Q_2, Q_3,$ and $Q_4$ are also very efficient in this database.

### 4.4. Visualization

Generalized spatial functions can represent information stored in the database as well as information contained in the output of a query. Here are some examples. The aforementioned query $Q_4$ (to find segments of space-time for a given temperature range $t_1$ to $t_2$) produces a generalized spatial function. The identity query $Q_0$ copies an entire stored spatial function into output. Query $Q_5$ produces the temperature function $T_{Carib} : R^4 \rightarrow R$ for a given observation source s. Query $Q_6$ produces source-averaged surface temperature data for the Caribbean Sea surface, $T_{Carib} : R^3 \rightarrow R$. Query $Q_7$ produces both surface temperature and ozone thickness data, (Temperature,Ozone): $R^3 \rightarrow R^2$. Query $Q_8$ produces a discrete spatial function which for each city and each month gives the monthly average temperature, Citytemp: $Cities\times\{1..12\} \rightarrow Temperature$. Query $Q_9$ produces a function $f_8 : Cities \times Time \rightarrow Temperature \times Ozone \times Factsets$. This function, for every city c and every moment in time t, gives the temperature, the ozone thickness, and the set of facts describing the events that were happening in the city at time t (i.e. events that started at or before t and ended after t). Here, Cities is a discrete domain and Time is a continuum. Query $Q_{10}$ produces uninterpreted SeaWiFS measurements in discrete orbital coordinates, together with information on satellite and lens positioning when making the measurements:

$$SW: OrbitPropagation \times Scan \times Time \times Band \rightarrow Color \times Factsets$$

In this section we discuss two methods we employ for visualization of such query results. The first one, animation, in not conceptually novel, but is interesting especially in
terms of performance ramifications in our system, as well as certain presentation enhancements that we introduce. The second, novel, method is the 3-D function superimposition.

4.4.1. Animation

Consider a query whose output is a three-dimensional function $f: \mathbb{R}^3 \rightarrow \mathbb{R}$, e.g. temperature (latitude, longitude, time).

We can display this function by mapping any two of the dimensions on the screen and translating the third dimension into a frame sequence. This can be seen as a movie with VCR-like controls — speed, pause, direction, rewind, etc. — as well as with zoom control.

Our systems also provides the user with pull-down menus or buttons to dynamically select a geographical display projection type for spatial functions. The projection types currently supported are: mercator, homographic, stereographic, sinusoidal, orthogonal, and orthographic.

To accommodate factsets in the visual presentation, e.g. the factsets produced in queries $Q_9$ and $Q_{10}$, we will be able to perform the following: when the user clicks at a point on the screen the system displays facts concerning that point.

When a query produces a spatial function of more than 3 dimensions, e.g. $f: \mathbb{R}^5 \rightarrow \mathbb{R}$, the viewing user will have buttons to dynamically freeze any dimensions and select two screen dimensions and one frame-sequence dimension.

Efficient support of animation requires very fast execution of spatial queries by the database server and very fast delivery of the results to the user. In our prototype, a cluster of database server machines is connected to user workstations via an ATM network, capable of simultaneously delivering 150 megabits per second to each user. The animation was implemented by David Barton, Elma Alvarez, and Martha Gutieruz. Illustrations 1 through 11 show snapshots of the user’s screen.

4.4.2. Function superimposition

Here, we would like to describe a novel visualization method that we employ: spatial function superimposition.

Consider a query whose output consists of two spatial functions of the same two-dimensional subspace. Example: the ozone layer thickness and the ocean temperature of a particular region on a particular date. The user posing this query desires to see correlation between the two functions. In visualization of this query we represent this output as a virtual reality 3-dimensional image, where the first function is mapped into elevation and the second into color. In our prototype system, the user can explore this image using virtual reality goggles: the user wearing such goggles can look at the image at various angles and positions. The computer senses the angle and the position of the person viewing the image via infra-red signaling, which the system uses to display appropriate views in relation to the user’s position. The depth illusion that the system implements works by displaying a stereo image: two quickly alternating images, each of which corresponds to the particular image that each eye would see if the query visualization were a 3-dimensional object in the real world. The goggles complete the effect by synchronously blocking the view of one of the eyes with an
LCD lens, fooling the brain into thinking that it is a real 3-dimensional object. A user without such goggles can still benefit from this data visualization method by rotating the image on the screen and looking at various pseudo-3D projections — an example of this is in the last Illustration 12.

Consider now a query whose output is a pair of spatial functions of a three-dimensional subspace \((F: R^3 \rightarrow R^2)\). Example: the ozone layer thickness and temperature of an ocean region during a given time interval — a query posed by a user interested to see how the temperature/ozone correlation changes in time. We present \(F\) as a sequence of functions \(F_t: R^2 \rightarrow R^2\). For each \(t\), \(F_t\) is viewed as a 3-D relief. The sequence of images \(F_t\) is viewed as an animated video with VCR-like controls (speed, play forward, rewind, frame by frame viewing, etc.), as well as with image controls (zoom, viewing angle, etc.).

Illustration 12 depicts \(F_0\), a snapshot of \(F\) at a time \(t_0\) — we froze the time to render a static 3-D image. In this example, the continents are colored brown, the ocean is colored from blue to red according to the surface temperature, and the ozone thickness is mapped into the relief's elevation. The ozone layer thickness happened to be measured when the hole in the ozone layer is apparent in the Antarctic region, as seen by a sharp slope at the left side of the snapshot. (This is a snapshot of a relief built on top of an orthogonal projection of the Earth; the image has been rotated and put in a perspective as if viewed from the Pacific Ocean westwards — this way the relief is clearly seen even without the help of the aforementioned 3-D goggles.)

The superimposition visualization program was written by Louis Florit. It is a part of the query visualization subsystem of our prototype semantic spatial database management system.

5. CONCLUSION

Our prototype for a massively parallel database management system is designed to efficiently realize complex queries on multi-dimensional spatial-temporal data and efficiently deliver the results to the user in an intelligible visual perspective. These direct and inverse relations between varied measures and geolocated objects are integral to deciphering and understanding Nature. The multi-dimensional visualization of highly intricate relations over massive amount of data aids in the rapid interpretation of complex and extended measures by domain scientists. We are certain that these features of our system are indispensable in effectively analyzing the enormous and imposing amount of scientific data quantifying our environment.

Acknowledgment.

We would like to thank the following persons for their help and advice: Artyom Shaposhnikov, Louis Florit, Elma Alvarez, Martha Gutierrez, Patrick Coronado, Keith Wichmann, Gyorgy Fekete, Kumar Krishen, Jerry Elliot, Robert Seals, David Buker, David Hislop, and Jagdish Chandra.
References


<table>
<thead>
<tr>
<th>Projection</th>
<th>Data</th>
<th>Rotation</th>
<th>Date</th>
<th>Cloud</th>
<th>Exit</th>
<th>Temperature in °C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orthographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orthogonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sinusoidal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stereographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Homolographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

-2
-1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

Date: 7/11/1987
<table>
<thead>
<tr>
<th>Projection</th>
<th>Data</th>
<th>Rotation</th>
<th>Date</th>
<th>Cloud</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orthographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orthogonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sinusoidal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stereographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Homolographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Ozone Layer in Dobson Units**

- < 113
- 114..125
- 126..137
- 138..149
- 150..161
- 162..173
- 174..185
- 186..197
- 198..209
- 210..221
- 222..233
- 234..245
- 246..257
- 258..269
- 270..281
- 282..293
- 294..305
- 306..317
- 318..329
- 330..341
- 342..353
- 354..365
- 366..377
- 378..389
- 390..401
- 402..413
- 414..425
- 426..437
- 438..449
- 450..461
- 462..473
- 474..485
- 486..497
- 498..509
- 510..521
- 522..533
- 534..545
- 546..557
- 558..569
- 570..581

Date: 9/21/1992
<table>
<thead>
<tr>
<th>Projection</th>
<th>Data</th>
<th>Rotation</th>
<th>Date</th>
<th>Cloud</th>
<th>Exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orthographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orthogonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sinusoidal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stereographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Homolographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ozone Layer in Dobson Units

Date: 9/23/1992

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;113</td>
<td>114</td>
<td>126</td>
<td>138</td>
<td>150</td>
<td>162</td>
</tr>
<tr>
<td>174</td>
<td>186</td>
<td>198</td>
<td>210</td>
<td>222</td>
<td>234</td>
</tr>
<tr>
<td>246</td>
<td>258</td>
<td>270</td>
<td>282</td>
<td>294</td>
<td>306</td>
</tr>
<tr>
<td>318</td>
<td>330</td>
<td>342</td>
<td>354</td>
<td>366</td>
<td>378</td>
</tr>
<tr>
<td>390</td>
<td>402</td>
<td>414</td>
<td>426</td>
<td>438</td>
<td>450</td>
</tr>
<tr>
<td>462</td>
<td>474</td>
<td>486</td>
<td>498</td>
<td>510</td>
<td>522</td>
</tr>
<tr>
<td>534</td>
<td>546</td>
<td>558</td>
<td>570</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Projection</td>
<td>Ozone Layer in Dobson Units</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orthographic</td>
<td>&lt; 113</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orthogonal</td>
<td>114 125</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sinusoidal</td>
<td>126 137</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stereographic</td>
<td>138 149</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Homolographic</td>
<td>150 161</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercator</td>
<td>162 173</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>174 185</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>186 197</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>198 209</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>210 221</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>222 233</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>234 245</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>246 257</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>258 269</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>270 281</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>282 293</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>294 305</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>306 317</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>318 329</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>330 341</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>342 353</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>354 365</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>366 377</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>378 389</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>390 401</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>402 413</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>414 425</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>426 437</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>438 449</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>450 461</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>462 473</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>474 485</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>486 497</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>498 509</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>510 521</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>522 533</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>534 545</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>546 557</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>558 569</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>570 581</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Projection</td>
<td>Data</td>
<td>Rotation</td>
<td>Date</td>
<td>Cloud</td>
<td>Exit</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>----------</td>
<td>------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>Orthographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orthogonal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sinusoidal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stereographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Homolographic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mercator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Date: 3/8/1995

SeaWiFS
Projection: Orthographic, Orthogonal, Sinusoidal, Stereographic, Homolographic, Mercator

Date: 3/6/1995