Towards a Modeling Framework for Geoscientific Data *

Eddie C. Shek†, Richard R. Muntz
Data Mining Laboratory
Computer Science Department
University of California, Los Angeles, CA 90024
{shek, muntz}@cs.ucla.edu

June 16, 1994

Abstract

Geoscience studies produce data from various observations, experiments, and simulations at an enormous rate. However, current database systems fail to provide the necessary combination of expressiveness, ease of use, flexibility, and efficiency to effectively support the analysis of complex spatio-temporal geoscientific datasets.

In this paper, we propose a data modeling framework which captures some of the important semantics of geoscientific data. The central concept behind the *field model* is that of the *field*, which is the association of points in an *coordinate space* with dependent data values from a *variable space*. A large variety of data fields can be modeled with index and variable spaces varying semantics. Semantic properties that are captured in the model include the ordering and regularity of elements in the index space. These, in turn, influence the appropriate data storage, indexing, and query optimization strategies.

In addition to being very large in size, geoscientific datasets are often sparse, irregular, and incomplete, due to noise and limitations imposed by sensors and communication equipment. As a result, it is sometimes necessary to derive dependent variable values from existing values during data analysis. The field model satisfies this requirement by allowing an *interpolation function* to be associated with a data field. And by tightly integrating it into the model, the query optimizer can take into account such on-the-fly derivation of data when it generates the query execution plan.

We have prototyped a parallel dataflow execution engine for queries expressed in the model. It is an important component of *QUEST*, a prototype data analysis environment providing content-based access to massive geoscientific datasets that is under development at UCLA.

---

*This research supported by NASA under HPCC grant NAG-2225.
†Also with Information Sciences Laboratory, Hughes Research Laboratories, Malibu, CA 90265.

1 Introduction

Geoscience studies produce data from various observations, experiments and simulations at an enormous rate. For example, NASA EOS expects to produce over 1 Tbyte of raw data and scientific products per day by the year 2000, and a 100-year UCLA AGCM simulation [30] generates approximately 30 Tbytes of output. However, often only a small percentage of the archived data is ever analyzed. Partly this has been due to the unavailability of sufficient storage and bandwidth at reasonable cost, and partly because of the inadequacies of traditional database management systems. In particular, current database systems fail to deliver the necessary combination of expressiveness, ease of use, flexibility, and efficiency to effectively support the analysis of complex spatio-temporal geoscientific datasets. Relational DBMSs are not expressive enough to naturally model complex scientific data, while object-oriented DBMSs have to be "tailored" to take advantage of the special semantics and structure of scientific data during query processing.

In this paper, we propose a modeling framework which captures several unique characteristics of geoscientific data, and hence provides the basis for exploiting these characteristics during query optimization. We also define an extensible algebra for the model; new complex user-defined operators can be integrated into the complete query optimization and processing framework. The *field model* serves as a foundation on which our further research on query optimization and parallelization, storage structures, and high performance I/O in geoscientific data analysis applications is based. In particular, geoscientific data:

- **are often measured over a continuous space.** The central idea behind the field model is that of the *field*, which is the logical association of points in an *coordinate space* with dependent data values from a distinct *variable space*. For instance, a sea-level pressure array is a field which have a 3-dimensional coordinate space (time-latitude-longitude), in which each "point" in a regular grid is associated with a pressure value (floating point number).

  When the coordinate space is continuous or infinite, it is impossible to explicitly record the variable value at every point of the coordinate space. The field model solves the problem by associating an *interpolation function* with each data field, allowing scientists to sample the coordinate space and record only a finite subset of the data, while making it possible to derive the rest of the data by implicitly storing them in the form of a function. By tightly integrating the concept of data interpolation into the model, the query optimizer can take into account the derivation of data when it generates the query execution plan. An extreme example of interpolation would be to allow geoscientific simulation models, e.g., UCLA AGCM, to be stored as the interpolation function, while only storing infrequent checkpoints as basis of "interpolation".

- **are often sparse, irregular, and incomplete.** Noise and limitations imposed by sensors and communication equipment frequently cause imperfections in observation data. An interpolation function associated with a data field is often necessary to derive missing dependent variable values from recorded ones during data analysis.

- **are extremely large in size.** Due to the extremely large size of scientific datasets, the set of bindings between independent points in the coordinate space and dependent variable values are often physically decomposed or "tiled" for parallelization. For example, a sea-level pressure field can be partitioned and stored as 2 fields covering the northern and southern hemisphere respectively. This kind of detail should be hidden from the scientists so that they can focus on the "science" instead of the data management issues. Nonetheless, it is captured in the physical data model so that it can be taken into account during query optimization.

- **are spatio-temporal in nature and include diverse data types.** Very different types of spatio-temporal geoscientific data ranging from multidimensional arrays of floating point numbers (e.g., sea-level pressure fields) to time series of points (e.g., cyclone tracks) need to be brought together and processed in a data analysis application. Spatial (point, line, polyline, polygon), temporal (time, relative time), and alpha-numeric (integer, floating point numbers, character strings, etc.) data types are included in the base type system in our implementation of the field model. Moreover, the model allows complex geoscientific data types to be defined by combining existing data types through the field construct. Furthermore, an ordering can be specified for data items in a field, allowing sequence queries to be issued against them.

- **have application-dependent operations that are computationally expensive.** The study of geosciences involves an elaborate process of exploratory data analysis. No one (not even the scientists) knows precisely the set of operators that are needed. Instead of having scientists develop applications on top of our model, we allow them to extend the model by introducing new functions and operators (e.g., optic flow algorithm for cyclone tracking) to the model which can then participate in the processing of query optimization.

- **often require content-based access.** The field model supports both "raw" field data (e.g., raster images) and complex geoscientific features (e.g., cyclone tracks are content-based indexes to the sea-level pressure field.) in a uniform data model and query processing framework. It raises the possibility of efficient use of geoscientific features as sophisticated content-based indexes to raw data fields.

In the next section, we outline some work in related areas that have influenced the development of our model. The field model is defined in section 3. Section 4 defines the field algebra and illustrates its use through example queries. Section 5 discusses some query processing issues related to geoscientific data management within the field model framework. Finally, section 6 concludes the paper with a brief summary of the field model.

2 Related Work

The unique data management requirements of scientific applications have been widely recognized [10, 11, 39, 40]. These scientific data modeling and management problems in several areas have been addressed by researchers. In this section, we outline some research that has influenced our approach to geoscientific data management.

2.1 Scientific Data Modeling

Scientific data analysis and visualization systems, e.g., AVS [1], IDL [33], Data Explorer [25], and Khoros [26], are widely used in the scientific community. Their comprehensive suites of mathematical and data analysis operators, in addition to visual dataflow programming environments allow scientists to conveniently manipulate, analyze, and visualize different types of scientific data. However, most of them implement operators as atomic units of operation and fail to provide a true dataflow processing environment. As a result, performance is sluggish for large datasets due to the lack of parallelism and the need to store large amounts of intermediate data.

Some work has been done in developing a data model for scientific visualization [12]. Most notably, the differential geometry notion of *fiber bundle* has been recognized by several researchers as an useful abstraction for an object-oriented scientific visualization data model [4, 5, 23, 28]. In brief, a fiber bundle is a space constructed by attaching a copy of a fiber space to each point in a base space. The field model borrowed this concept of space "bundling". However, research in scientific visualization modeling concentrates mainly on the mathematical basis of data models and typically ignores physical data management issues like query optimization, parallelization, and storage layout.

2.2 Complex Data Support

Relational databases require that relations be at least in first normal form. However, the requirement of atomic-valued domains makes it difficult to support some non-traditional applications like scientific data analysis. To overcome the restrictions of 1NF relational model, researchers have proposed various extensions. Nested relational models [29, 35, 37] model complex objects by allowing a relation to have relation-valued attributes. Relationships between data items can be explicitly represented by the nesting and hence do not have to be realized by applying expensive join operations. We have adopted some of the concepts developed in non-1NF relational data models in the field model (e.g., the nest and unnest operators). Other extensions to the relational model that are important to scientific applications include sequenced data and list-valued attributes [15, 19, 20, 32, 34].

Object-oriented DBMSs and extensible relational DBMSs with flexible type systems are gaining in popularity because of their expressiveness; they are much better suited for scientific applications than RDBMSs. Unfortunately, current systems fail to satisfy some of the requirements of scientific data management [27]. For example, OODBMSs have problems with the retrieval of small segments of data within a large data object. More importantly, while *encapsulation* allows the implementation of complex object types to be hiden from the user, it also hides the structure and semantics of the type from the query optimizer. As a result, it is difficult for the optimizer in OODBMSs and extended RDBMSs to exploit the special semantics and structures of scientific data to generate efficient query execution plans. This imposes a severe limitation to the use of such systems for scientific applications considering the size of the data and the complexity of the applications. We try to overcome this limitation by explicitly capturing structural and semantic information about the scientific data in the field data model.

3 The Field Model

In this section, we present the *field model*, a data modeling framework for geoscientific data management. A goal of the field model is to provide a rich but conceptually simple data model to encompass the wide variety of scientific data types, while capturing significant semantic and structural properties that have implications on the processing of such data. We concentrate on the logical model in this section; we will discuss physical modeling issues that arise as a result of our logical modeling decisions in a later section.

3.1 Semantic and Structural Properties of Geoscientific Data

A field captures several important semantic and structural properties of geoscientific data that have significant implications on the appropriate choice of storage, indexing, and query optimization strategies.

3.1.1 Separation of Coordinates and Variables

Scientific data often come in the form of data fields in which each point in a multi-dimensional spatio-temporal *coverage* is assigned a value from a variable space. Each point in the coverage of a field belongs to the field's coordinate space. Some simple spaces in which a scientist may be interested, include temporal spaces like absolute time $\mathcal{T}$, relative time $\mathcal{T}_r$, time interval $\mathcal{TI}$, and georeferencing spatial spaces like latitude $\mathcal{LAT}$, longitude $\mathcal{LON}$, and altitude $\mathcal{ALT}$. They can be put together to form more complex composite spaces, e.g., longitude and latitude combined form a space at the same altitude over the surface of the earth.

Note that in general, a "point" (or coordinate) in a coordinate space is not necessarily a point in the usual sense. For example, an integer is a "point" in the integer space (domain) $\mathcal{N}$. At the same time, we can define an abstract data type of $1° \times 1°$ tiles over the surface of the earth. Each "point" in a space containing such tiles is a $1° \times 1°$ area.

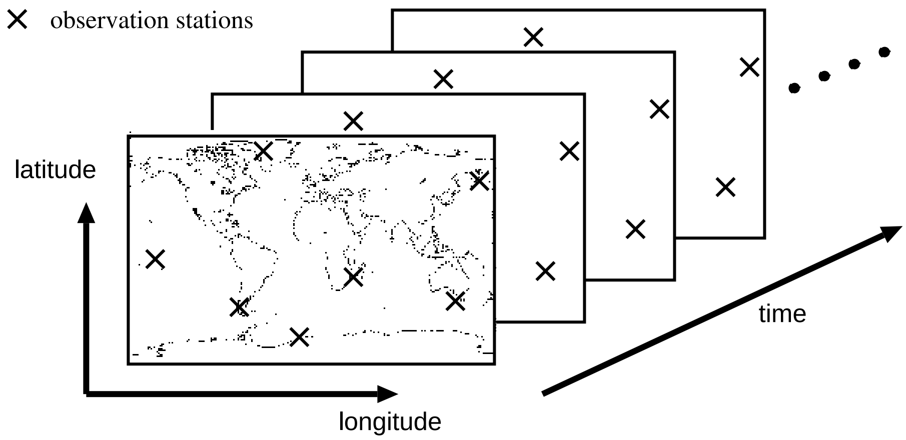

Figure 1: Observation stations

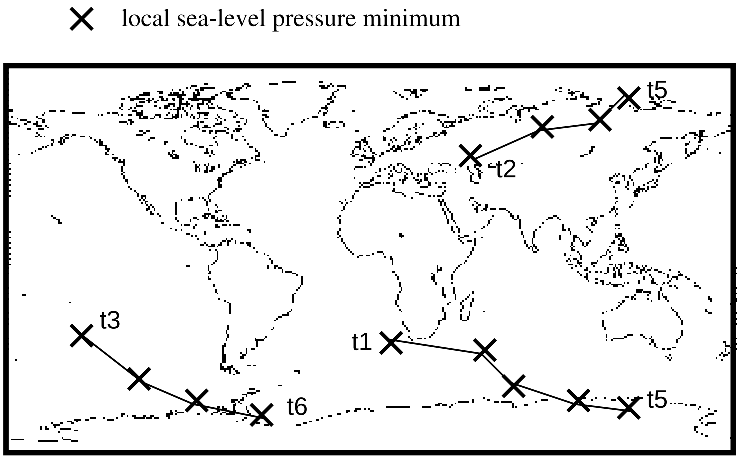

Figure 2: Cyclone tracks

Consider a set of fixed observation stations on the earth which take measurements at fixed intervals (see figure 1). The coordinate space in this case is three dimensional, and is irregular in the two spatial dimensions but regular in the time dimension. A finite set of points specified by longitude and latitude may specify the location of observation sites. A regularly spaced coordinate set in the time dimension might be used to indicate daily measurements. As another example, consider a type of physical phenomenon which moves over the surface of the earth and for which we can assign a reference point (its center) at each time instant (see figure 2). Further assume that we observe and record the location of the center for each instance of the phenomenon to form trajectories or tracks. Here again we have three dimensional data but in this case each trajectory can be viewed as a time varying location. For each trajectory the coordinate space is time and the value at a specific time is the spatial coordinates (an possibly other attributes) of the phenomenon recorded at that time instant.

3.1.2 Intensional Data

Geoscientific datasets are extremely large in size. They may even have continuous or infinite coverages. It is sometimes impossible or impractical to explicitly store data values at all points in the coverage. As a result, we make a distinction between explicitly stored and "intensional" field data. We define a finite subset of the coverage called the *sampled coverage* to be the set in which the mapping from each coordinate to a variable value is explicitly stored, possibly in a lookup table. On the other hand, the variable values associated with the rest of the coordinates in the coverage are derived from the variable mappings by an *interpolation function*. Note that there may be a number of different logical fields sharing the same set of stored data but with different interpolation functions. In addition, geoscientific data are often sparse, irregular, and incomplete. For example, observation stations are sparse and irregular. An interpolation function allows a more structured view of the data to be presented to the scientists and hence simplifies queries and applications programs. Figure 3 shows several interpolation functions with different semantics for a time series [38].

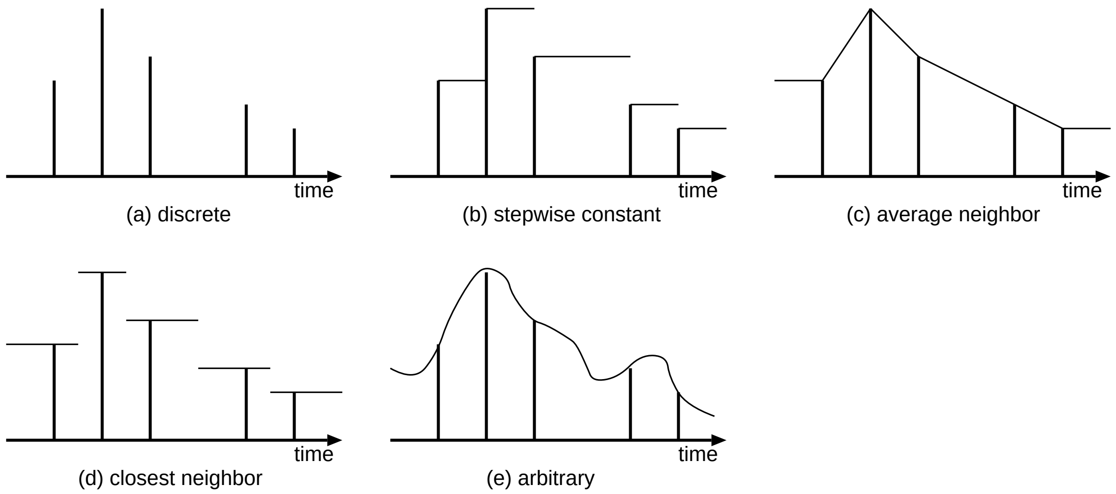

Figure 3: Interpolation functions for a time series

Theoretically, the interpolation function can be arbitrarily complex and take additional arguments, including other fields. For example, UCLA AGCM output can be represented in the field model by only storing the checkpoint data values and having the complete model behave like an interpolation function, i.e., deriving any requested data by initializing the model with the most recent checkpoint and running the model forward.

However, selection of coordinates in the coverage of a field based on variable values derived from a complex interpolation function may not be computable in finite time. Consider a sea-level field with a complex curve-fitting interpolation function. In order to find the constant-pressure contour at the sea-surface at an infinite resolution, we need to test the pressure at every point in the continuous and infinite coverage and the contour will not have a finite extensional representation, unless we are provided with the appropriate contouring function that matches the curve-fitting function. Given a field with complex interpolation function, operations on it that are dependent on derived variable values can be handled by user-defined operators that are specific to the interpolation function.

3.2 Definitions

We define a *field space* $[C, V]$ as a data type in which each instance is a data field mapping variable values of type V to points in a subset of the points belonging to coordinate space C. The field space defines a logical association (or bundling) of the coordinate space C with the variable space V. Formally, a field F belonging the field space $[C, V]$, $F \in [C, V]$, is defined as

$$F = \langle\ U, S, v\ \rangle$$

where

- the *coverage* $U \subseteq C$ is the subset of the coordinate space for which each coordinate is associated with a variable value, either explicitly or implicitly.
- the *sampled coverage* $S \subseteq U$ is the *finite* subset of the coverage in which each coordinate is *explicitly* associated with a value in the variable space.
- a *variable mapping* $v : U \rightarrow V$ maps a variable value to each coordinate in the field's coverage. Each coordinate in the sampled coverage is explicitly mapped to a variable, for example by means of a lookup table. On the other hand, an *interpolation function* is used to derive variable values for points in the part of the coverage excluded from the sampled coverage.
- a *coordinate record* $\langle p, v(p)\rangle$ at a coordinate point in the field's coverage $p \in U$ is defined as the tuple formed by the coordinate and its associated variable value. This is a useful concept when we want to refer to both a coordinate point and the variable value associated with it in the same context together as a single data item.
- a *neighborhood* is a set of coordinate records, each of which bears a specified relationship to a particular coordinate record called the neighborhood *focus*. A neighborhood is defined by a predicate on a pair of coordinate records (either from the same field or different fields). Given a neighborhood predicate $\theta : (C_1 \times V_1) \times (C_2 \times V_2) \rightarrow \mathcal{BOOL}$ and a focal coordinate record $f \in (C_1 \times V_2)$, a neighbor coordinate record $r \in (C_2 \times V_2)$ lies in the neighborhood of f if $\theta(f, r) = true$.

Similar to the relational model, we allow the dimensions of the coordinate and variable spaces to be named in order to facilitate the expression of operations on a field. A field with an m-dimensional coordinate space and an n-dimensional variable space can be specified as

$$F(a_1^I, a_2^I, \ldots, a_m^I; a_1^V, a_2^V, \ldots, a_n^V)$$

where $a_1^I, a_2^I, \ldots, a_m^I$ are the names of the index attributes and $a_1^V, a_2^V, \ldots, a_n^V$ are the names of the variable attributes respectively. Furthermore, we deonote the sampled coverage, coverage, and variable mapping of the field as S_F, U_F, and v_F respectively.

3.3 Example Field Types

The field model is a very powerful data model. Complex data structures can be modeled by combining different coordinate and variable spaces with different semantics and structures. We will illustrate the different field types with example fields from an atmospheric science application.

3.3.1 Arrays

Multi-dimensional arrays constitute one of the most common type of fields. An array is a field with a regular and discrete coverage composed of the array ranges in the different dimensions. Data values for points in the field's coverage are usually explicitly stored, in which case the sampled coverage is equivalent to the coverage.

3.3.2 Continuous Data Fields

An interpolation function is needed when a data field has a continuous coverage which is sampled. Consider the sea-level pressure field $SLP(t, lon, lat; slp) \in [\mathcal{T} \times \mathcal{LON} \times \mathcal{LAT}, \mathcal{R}]$ (see figure 4). t, lon, and lat are the names of the coordinate attributes time, longitude, and latitude respectively, while slp is the name of the variable attribute, sea-level pressure value.

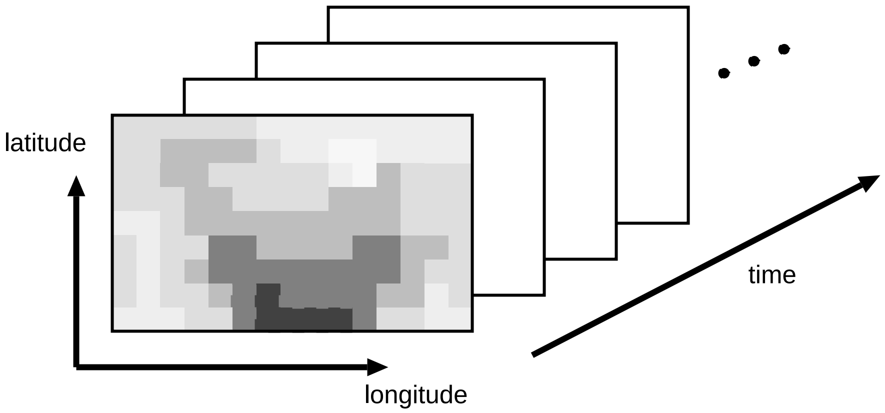

Figure 4: Sea-level pressure field

Assuming that the sea-level pressure is recorded everyday at noon from 1/1/1980 to 12/31/1989 over regular 5° longitude by 4° latitude grids over the surface at the earth, the sampled coverage of the sea-level pressure field S_{SLP} is a finite, regular subset of the coordinate space

$\mathcal{T} \times \mathcal{LON} \times \mathcal{LAT}$ defined as

$$S_{SLP} = \left\{ \begin{array}{lcl} \langle t, lon, lat \rangle \in \mathcal{T} \times \mathcal{LON} \times \mathcal{LAT} & | & t \text{ is noon from } 1/1/1980 \text{ to } 12/31/1993 \\ & & \wedge\ lon \in \{-180, -175, -170, \ldots, 175\} \\ & & \wedge\ lat \in \{-88, -84, -80, \ldots, 88\} \end{array} \right\}.$$

A subset of the field's variable mapping v_{SLP} is shown in table 1. The interpolation function, for example, might assign to each index point in the coverage the value of the closest sampled index point.

t	lon	lat	slp
1/1/93	0°	0°	975
1/1/93	5°	0°	972
1/1/93	10°	0°	977
1/1/93	0°	4°	969
1/1/93	5°	4°	971
1/1/93	10°	4°	972
1/1/93	0°	8°	965
1/1/93	5°	8°	970
1/1/93	10°	8°	969

Table 1: A subset of the variable mapping of sea-level pressure field SLP

A temperature field $TMP(t, lon, lat; tmp) \in [\mathcal{T} \times \mathcal{LON} \times \mathcal{LAT}, \mathcal{R}]$ can be defined similar to SLP.

3.3.3 Time Series

A time series is a special case of sequenced data and is naturally modeled by a field with a temporal index space. An example time series is a cyclone track which can be modeled as a point whose location changes with time. We can define a field $A\_CYC(t; SLP\_MAP(lon, lat; slp)) \in [\mathcal{T}, [\mathcal{LON} \times \mathcal{LAT}, \mathcal{R}]]$ to store a cyclone track as a time series of points at sea level and local pressure minima. The coverage U_{A_CYC} is the lifespan of the cyclone, and the sampled coverage S_{A_CYC} is the set of times at which sea-level minima on the trajectory of the cyclone are measured. Each time in the coverage is associated with a field consisting of 1 mapping from the location of the cyclone at that time to the sea-level pressure. The interpolation function "connects" the sampled coordinates on the cyclone track. It either interpolates the sea-level pressure values along the track or retrieves them from the sea-level pressure field. Table 2 shows the variable mapping of a possible cyclone track field.

t	SLP_MAP		
	lon	lat	slp
1/1/93	0°	0°	975
1/2/93	5°	4°	972
1/3/93	5°	4°	978
1/4/93	10°	0°	974

Table 2: The variable mapping of the cyclone track field A_CYC

3.3.4 Sets, Relations, and Multisets

A set (or a relation without duplicated tuples) can be represented in the field model as a field with an empty variable space. A set of points in a space S is equivalent to a field belonging to the field space $[S, \emptyset]$ with sampled coverage S. We denote the field space as $\{S\}$. The sampled coverage of a set is precisely the set of data items. Since all data items have to be explicitly stored, the coverage and the interpolation function are equivalent to the sampled coverage and the variable mapping respectively. For example, the set of integers $\{1, 2, 3\}$ can be specified as the field $\langle\ \{1, 2, 3\}, \{1, 2, 3\}, \emptyset\ \rangle$.

On the other hand, a multiset of data items from space S can be modeled as a field in field space $[S, \mathcal{N}]$ where each data item in the multiset is associated with the number of times it appear in the multiset. We define the notation for the multiset as $\{\{S\}\}$.

Based on the cyclone track field A_CYC, we can define a set of cyclone tracks as a nested field $CYC\_SET\{A\_CYC(t; SLP\_MAP(lon, lat; slp))\} \in \{[\mathcal{T}, [\mathcal{LON} \times \mathcal{LAT}, \mathcal{R}]]\}$. The coverage and sample coverage are both the set of explicitly stored cyclone track fields.

Alternatively, if cyclone tracks have unique integer cyclone ids, the set of cyclone tracks can be specified as $CYCS(id; A\_CYC(t; SLP\_MAP(lon, lat; slp)) \in [\mathcal{N}, [\mathcal{T}, [\mathcal{LON} \times \mathcal{LAT}, \mathcal{R}]]]$. The coverage and sampled coverage both equal the set of cyclone ids and each cyclone id gets mapped to a cyclone track field by the variable mapping.

4 Field Algebra

In the previous section we defined the field data model. In this section, we describe the set of operators defined for the field model. We should emphasize that they are applicable to all fields. The operators are organized into the following classes:

- Relational(-like) operators similar to those in the relational algebra.

- Space conversion operators which derive new fields with different coordinate and variable spaces. Users can extend the model by introducing application-specific space conversion functions.

- Grouping operators which collect related coordinate records in a field for further processing.

- Sequenced field selection operator which selects data items based on their ordering in a field.

Note that the algebra presented in this paper is not meant to be minimal. Instead, a large suite of operators is defined to allow scientists to conveniently express their intentions.

4.1 Relational Operators

We begin the specification of the field algebra by describing the set-oriented (nested) relational operators. The definitions of selection, projection, cartesian product, union, intersection, difference, and join operators are similar to their formulations in the relational algebra. They are necessary for the field model to support the capabilities of the relational model.

4.1.1 Selection σ

The selection operator σ restricts the coverage of a field to the set of coordinate points satisfying a selection condition. Selection returns a field in which points in the coverage lie in a neighborhood of a specified focal point. The selection condition, expressed in terms of a neighborhood predicate and a focal point, can essentially be any predicate defined on the coordinate and variable attributes of the field. Note that as described in the last section, selections based on the variable values of a field with complex interpolation function should be handled by using user-defined operators specific to the interpolation function, due to the uncomputability and infiniteness of representation of the result.

Selection on a field $F \in [C_F, V_F]$, given neighborhood predicate $\theta : (C_F \times V_F) \times (C_F \times V_F) \rightarrow \mathcal{BOOL}$, and focal point $f \in (C_F \times V_F)$ is defined as

$$\sigma_{\theta,f}(F) \;=\; \langle\, U, S, v \,\rangle$$

where

$$\begin{aligned} U \;&=\; \{\, p \in U_F \;\mid\; \theta(f, \langle p, v_F(p)\rangle) \;=\; true \,\}, \\ S \;&=\; \{\, p \in S_F \;\mid\; \theta(f, \langle p, v_F(p)\rangle) \;=\; true \,\}, \\ &\forall p \in U,\; v(p) \;=\; v_F(p). \end{aligned}$$

4.1.2 Cartesian Product $\times$

The coordinate space and variable space of the cartesian product of fields F and G, denoted as $F \times G$, are the cartesian products of those of F and G respectively. The coverages of field F and G combine to form the coverage of their cartesian product. Similarly, the cartesian product of their variable spaces becomes the variable space of their cartesian product.

Given fields F and G, cartesian product is defined as

$$F \times G \;=\; \langle\; U, S, v \;\rangle$$

where

$$\begin{aligned} U &= U_F \times U_G, \\ S &= S_F \times S_G, \\ \forall p \in U_F, \forall q \in U_G,\; v(p) &= \langle v_F(p), v_G(q) \rangle. \end{aligned}$$

4.1.3 Union $\cup$, Intersection $\cap$, and Set Difference $-$

Given 2 input fields belonging to the same field space, the union operator collects variable values associated with each point in the union of their coverages. Because a point in the common coverage can be associated with different variable values in the input fields, each point in the union is associated with a multiset containing the variable values associate with it in the input fields.

Similar to union, the intersection and set difference operators collect variable values from coordinate points in input fields. The only difference is that the result coverages are the intersections and set differences of those of the input fields respectively.

Formally, Given fields $F_1, F_2 \in [C_F, V_F]$, for operator $\oplus \in \{\, \cup, \cap, - \,\}$,

$$F \oplus G \;=\; \langle\; U, S, v \;\rangle$$

where

$$\begin{aligned} U &= U_{F_1} \oplus U_{F_2}, \\ S &= S_{F_1} \oplus S_{F_2}, \\ \forall p \in U_{F_1} \oplus U_{F_2},\; v(p) &= \{\{\; v_{F_1}(p), v_{F_2}(p) \;\}\}. \end{aligned}$$

4.1.4 Unnesting ν and Nesting μ

In the field model, a nested field is one in which the values associated with points in the coverage are fields themselves. Consider a nested field $F \in [C_1, [C_2, V]]$. Unnesting the nested field F moves the nested variable space's coordinate space C_2 into the coordinate space of the field. The result of unnesting F is a field $\nu(F)$ belonging to the field space $[C_1 \times C_2, V]$.

To retrieve a variable value $x \in V$ from F, we need to go through 2 mappings: first from a point $p_1 \in U_F$ to a field $G = v_F(p_1) \in [C_2, V]$, then from another point in the coverage of G, $p_2 \in U_G$ to variable value $x = v_G(p_2) \in V$. On the other hand, we can retrieve the same variable value from $\nu(F)$ with only one mapping $x = v_{\nu(F)}(\langle p_1, p_2 \rangle)$. For example, table 3 shows that repeatedly unnesting the cyclone track field results in the set of all points in some cyclone track $CYCPT$.

id	t	lon	lat	slp
0	1/1/93	3°	1°	971
0	1/2/93	5°	0°	972
0	1/3/93	6°	2°	974
...	...	...	...	...

Table 3: $CYCPT(id, t, lon, lat; slp) \; = \; \nu(\nu(CYCS))$

Formally, given field $F \in [C_F, [C_{V_F}, V_{V_F}]]$,

$$\nu(F) \; = \; \langle \; U, S, v \; \rangle$$

where

$$U \; = \; \{ \; \langle p, q \rangle \; \mid \; p \in U_F \; \wedge \; q \in U_{v(p)} \; \},$$

$$S \; = \; \{ \; \langle p, q \rangle \; \mid \; p \in S_F \; \wedge \; q \in S_{v(p)} \; \},$$

$$\forall p \in U_F, \; q \in U_{v(p)}, \; v(\langle p, q \rangle) \; = \; v_{v(p)}(q).$$

The nest operator μ has the opposite effect of the unnest operator. It moves selected coordinate attributes of a field into the variable space. Each point in the coverage of the resulting field is associated with a field whose coordinate space is composed of the migrated attributes. Table 4 shows the result of nesting the longitude and latitude attributes of the sea-level pressure field SLP.

Formally, nesting coordinate attributes $c_1, \ldots, c_k$ of a field $F(a_1, \ldots, a_i; b_1, \ldots, b_j)$ is denoted as

$$\mu_{c_1,\ldots,c_k}(F) \in [\pi_{d_1,\ldots,d_l}(C_F), [\pi_{c_1,\ldots,c_k}(C_F), V_F]]$$

where $d_1, \ldots, d_l \in \{a_1, \ldots, a_i\} - \{c_1, \ldots, c_k\}$ are the coordinate attributes that are not nested. The result of the nest operator satisfies the condition that

$$\nu(\mu_{c_1,\ldots,c_k}(F)) = F.$$

By combining the unnest and nest operators, we can freely move attributes between the coordinate and variable spaces. For example, given a field $F(a_1, \ldots, a_i; G(b_1, \ldots, b_j; c_1, \ldots, c_k))$,

t	SLP_FLD		
	lon	lat	slp
1/1/93	0°	0°	975
	5°	0°	972
	10°	0°	977
	...	...	...
1/2/93	0°	0°	970
	5°	0°	982
	10°	0°	975
	...	...	...
...	...	...	...

Table 4: $SLP\_NEST(t; SLP\_FLD(lon, lat; slp)) \; = \; \mu_{lon,lat}(SLP)$

we can move attributes the attribute b_1 (or any list of attributes in general) into the coordinate space of F by applying the unnest and nest operators to it in succession

$$\mu_{b_2,\ldots,b_j}(\nu(F)).$$

4.1.5 Natural Join $\bowtie$

Natural join is particularly useful in the field model. It naturally joins the coverages of 2 fields (probably in different field spaces). The variable value associated with each point in the result field's coverage is the tuple formed by the variable values of the corresponding input coordinate points. Given two fields with matching coordinate spaces, it is often useful to increase the dimensionality of the variable space using the natural join, combining the variable values associated with each point in the input fields' coverages. Table 5 shows the result of naturally joining the sea-level pressure and sea-level temperature fields.

Formally, given fields $F \in [C_F, V_F]$ and $G \in [C_G, V_G]$, natural join is defined as

$$F \bowtie G \;=\; \langle\, U, S, v \,\rangle$$

where

$$U \;=\; U_F \bowtie U_G,$$

$$S \;=\; S_F \bowtie S_G,$$

$$\forall p \in U_F, q \in U_G,\; v(p \bowtie q) \;=\; \langle v_F(p), v_G(q)\rangle.$$

t	lon	lat	slp	tmp
1/1/93	0°	0°	975	300
1/1/93	5°	0°	972	302
1/1/93	10°	0°	977	303
1/1/93	0°	4°	969	297
1/1/93	5°	4°	971	299
1/1/93	10°	4°	972	299
...	...	...	...	...

Table 5: $SLP \bowtie TMP$

4.1.6 Neighborhood Join $\bowtie_\theta$

In addition to natural join, we allow 2 fields to be joined based on more general join conditions. The join condition is defined as a neighborhood predicate defined on a pair of coordinate records, one from each of the input fields. Each coordinate record in the first input field is joined with all the coordinate records in the second input field which lie in its neighborhood. It is similar to θ-join in the relational algebra as the neighborhood function can be any predicate defined on the attributes of the input fields. For example, we can join the set of cyclone track points with sea-level temperature within a 5° radius. Table 6 shows the result of the join. However, similar to selection, joining 2 fields based on the variable values may not be computable if the interpolation functions of the fields do not have well know semantics. Such joins should be evaluated by user-defined operators that properly capture and handle the semantics of the interpolation functions of the input fields.

id	t	lon	lat	tmp_t	tmp_lon	tmp_lat	slp	tmp
0	1/1/93	3°	1°	1/1/93	0°	0°	971	300
0	1/1/93	3°	1°	1/1/93	5°	0°	971	302
0	1/2/93	5°	0°	1/1/93	5°	0°	972	302
0	1/2/93	5°	0°	1/1/93	0°	0°	972	300
0	1/2/93	5°	0°	1/1/93	10°	0°	972	303
0	1/3/93	6°	2°	1/1/93	10°	0°	974	303
0	1/3/93	6°	2°	1/1/93	5°	0°	974	302
...	...	...	...	...	...	...	...	...

Table 6: $CYCPT\_TMP(id, t, lon, lat, tmp\_t, tmp\_lon, tmp\_lat; slp, tmp) = CYCPT \bowtie_\theta TMP$ where $\theta \equiv CYCPT.t = TMP.t \wedge within\_5°(CYCPT.lon, CYCPT.lat, TMP.lon, TMP.lat)$

Given fields $F \in [C_F, V_F]$ and $G \in [C_G, V_G]$, and neighborhood predicate $\theta : (C_F \times V_F) \times$

$(C_G \times V_G) \rightarrow \mathcal{BOOL}$, neighborhood join is defined as

$$F \bowtie_\theta G \;=\; \langle\, U, S, v \,\rangle$$

where

$$\begin{aligned} U &= \cup \{\, \langle p, q\rangle \;\mid\; p \in U_F \wedge q \in U_G \wedge \theta(\langle p, v_F(p)\rangle, \langle q, v_G(q)\rangle) = true \,\}, \\ S &= \cup \{\, \langle p, q\rangle \;\mid\; p \in S_F \wedge q \in S_G \wedge \theta(\langle p, v_F(p)\rangle, \langle q, v_G(q)\rangle) = true \,\}, \end{aligned}$$
$$\forall p \in U_F, q \in U_G, \langle p, q\rangle \in U,\; v(\langle p, q\rangle) \;=\; \langle v(p), v(q)\rangle.$$

4.1.7 Coordinate Projection π^C and Variable Projection π^V

Since the coordinate and variable spaces are distinct and play different roles, the field model includes 2 projection operators: coordinate projection π^C and variable projection π^V for projections on the coordinate and variable spaces respectively.

Variable projection behaves similar to relational projection. After variable projection, each coordinate point in the coverage of a field is associated with a projection on the the original variable value. Projecting variable attributes $b_1, \ldots, b_l$ from a field F is defined as

$$\pi^V_{b_1,\ldots,b_l}(F) \;=\; \langle\, U_F, S_F, v \,\rangle$$

where

$$\forall p \in U_F,\; v(p) \;=\; \pi_{b_1,\ldots,b_l}(v_F(p)).$$

On the other hand, projecting dimensions from the coordinate space is more complicated. Projecting coordinate dimensions from a field results in another field with a reduced coordinate space in which each point is mapped to one or more variable values. The variable values associated with points in the input field's coverage that project into a point in the result coverage are collected in a multiset, which is the variable value of the point. Table 7 shows the result of projecting out coordinate dimensions related to temperature from the join of cyclone track points and temperature.

Projecting coordinate attributes $a_1, \ldots, a_k$ from a field F is defined as

$$\pi^C_{a_1,\ldots,a_k}(F) \;=\; \langle\, U, S, v \,\rangle$$

where

$$\begin{aligned} U &= \pi_{a_1,\ldots,a_k}(U_F), \\ S &= \pi_{a_1,\ldots,a_k}(S_F), \\ \forall p \in U,\; v(p) &= \{\{\, v_F(\pi_{a_1,\ldots,a_k}(q)) \;\mid\; \pi_{a_1,\ldots,a_k}(q) = p \,\}\}. \end{aligned}$$

Sometimes a user may want to perform projection on a "compound" attribute (e.g., projecting out the year and day from the time attribute in $CYCPT_TMPS$ in order to collect a multiset of sea-level pressure and temperature values for each month). We will describe how this can be done in the next section by means of space conversion operators. In short, we can apply a conversion function to each time value and map it into a month value.

id	t	lon	lat	$TMPS$	
				slp	tmp
0	1/1/93	3°	1°	971	300
				971	302
0	1/2/93	5°	0°	972	302
				972	300
				972	303
0	1/3/93	6°	2°	974	303
				974	302
...	...	...	...	...	...

Table 7: $CYCPT\_TMPS(id, t, lon, lat; TMPS\{\{\, slp, tmp\,\}\}) \;=\; \pi^{C}_{id,t,lon,lat}(CYCPT\_TMP)$

4.2 Space Conversion

A stumbling block of geoscientific research is the diversity and heterogeneity of data. Consider a 4-dimensional (time-longitude-latitude-altitude) pressure field. First of all, there are numerous coordinate systems, geographic projections, and georeferencing systems in which data can be recorded. At the same time, we can represent the data with the altitude as a coordinate and pressure as the value, or alteratively have pressure as a coordinate and the altitude at which the pressure is recorded as the value (assuming that pressure monotonically decreases with altitude). As a result, the coordinate and variable spaces of a field often need to be converted in order for the field to be compared to and processed with relevant data.

4.2.1 Resampling *resample*

The *resample* operator changes the sampling of a field and generates variable values with the field's interpolation function at a set of new sampled coordinate points. For example, a user has to resample the sea-level pressure field if he/she wants to study sea-level pressure at a different resolution from the one assumed by the original field. A particularly common use of the resample operator is to change the granularity of the sampled coverage. This can be used to "thin" or "thicken" a rectilinear coordinate space by changing the spacing between grid points in one or more dimensions. Figure 5 shows the effect of changing the granularity of a regular grid.

Given a field F and a desired sampled coverage $S \in U_F$ belonging to the field's coverage, *resample* derives a new field in which variable values are explicitly associated with points in S. Formally,

$$resample_S(F) \;=\; \langle\; U_F, S, v_F \;\rangle.$$

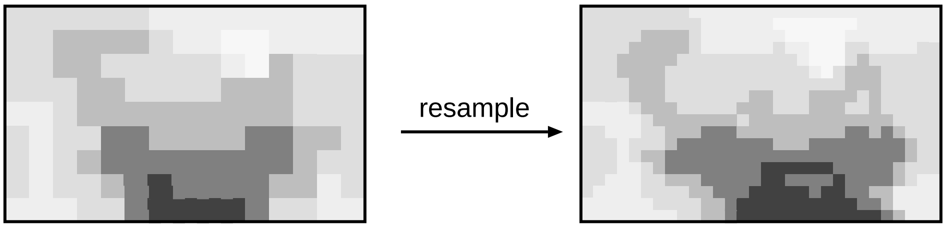

Figure 5: Resampling field over spherical surface with a finer grid size

4.2.2 Coordinate Space Conversion $convert^C$

A field's coordinate space can be changed by mapping each point in it into a point in another space (e.g., convert from sample number in an experiment to actual time, or convert one map projection to another). Alternatively, a user may not want to change the coordinate space but only "move" the coordinate points relative to the variable values (e.g. translation or rotation). The coordinate space conversion operator $convert^C$ allows the above mentioned operations on a field to be expressed.

Given a one-to-one mapping $\triangleright : C_F \rightarrow C$ from the coordinate space of field $F : [C_F, V_F]$ to another coordinate space C, coordinate conversion is defined as

$$convert^C_{\triangleright}(F) \;=\; \langle\, U, S, v \,\rangle$$

where

$$\begin{aligned} U \;&=\; \{\, \triangleright(p) \;\mid\; p \in U_F \,\}, \\ S \;&=\; \{\, \triangleright(p) \;\mid\; p \in S_F \,\}, \\ \forall p \in U_F,\; v(\triangleright(p)) \;&=\; v_F(p). \end{aligned}$$

On the other hand, if the coordinate conversion function $\triangleright$ is many-to-one, multiple coordinates in the input field can be mapped to a single coordinate in the result coverage. In this case, the multiset of variable values associated with the input coordinates is associated with the result coordinate to which the input coordinate is mapped. Note that the coordinate projection operator is a special case of coordinate conversion in which the coordinate function projects attributes from the coordinate space. Coordinate conversion is formally defined in this case as

$$convert^C_{\triangleright}(F) \;=\; \langle\, U, S, v \,\rangle$$

where

$$\begin{aligned} U \;&=\; \{\, \triangleright(p) \;\mid\; p \in U_F \,\}, \\ S \;&=\; \{\, \triangleright(p) \;\mid\; p \in S_F \,\}, \\ \forall p \in U,\; v(p) \;&=\; \{\!\{\, v_F(q) \;\mid\; q \in U_F \;\wedge\; p = \triangleright(q) \,\}\!\}. \end{aligned}$$

4.2.3 Variable Space Conversion $convert^V$

The variable space conversion operator $convert^V$ derives new variable values for all points in the coordinate space of a field by applying a user-defined function (or operator) to each coordinate record. Each coordinate point in the resulting field is associated with the value returned by the function when applied to the corresponding coordinate record in the original field. A use of variable space conversion is to perform aggregation on variable values after they are grouped into fields, sets, or multisets by operators that create collections of data values in the variable space of a field. Note that variable projection is a special case of variable conversion in which attributes are projected from the variable space. As an example, table 8 shows the result of computing the average sea-level temperature in a 5° region centered at each point on a cyclone track, after temperature values in the neighborhood of each point of a cyclone track are collected into a multiset in the variable space of field $CYCPT\_TMP$.

id	*t*	*lon*	*lat*	*slp*	*avgtmp*
0	1/1/93	3°	1°	971	301
0	1/2/93	5°	0°	972	301.7
0	1/3/93	6°	2°	974	302.5
...	...	...	...	...	...

Table 8: $CYCPT\_AVGTMP(id, t, lon, lat; slp, avgtmp) \ = \ convert^V_{avg}(CYCPT\_TMPS)$

Let $\triangleleft : C_F \times V_F \rightarrow V$ be a function that maps each coordinate record in field F to some value in V. Then

$$convert^V_{\triangleleft}(F) \ = \ \langle\, U_F, S_F, v \,\rangle$$

where

$$\forall p \in U_F, \ v(p) \ = \ \triangleleft(\langle p, v_F(p)\rangle).$$

4.3 Coordinate Record Collection

Data analysis applications often involve the generation of aggregate information on collections of related data from a field. We provide several tuple collection operators for collecting subfields containing related coordinate records.

4.3.1 Neighborhood Grouping *group*

The grouping operator *group* associates with each coordinate in a field's coverage a value equal to the subfield composed of all coordinate records in the field that lie in the neighborhood of that point. Grouping is usually done in preparation for an operator that will compute a value for each point based on values in the neighborhood of that point. For example, table 9 shows the field produced as a result of grouping all sea-level pressure points within 5° of each sampled coordinate point in a subfield of SLP. A function can that be applied to the field to calculate the average measured sea-level pressure in 5° regions around each point.

t	lon	lat	SLP_FLD			
			t	lon	lat	slp
1/1/93	0°	0°	1/1/93	0°	0°	975
			1/1/93	5°	0°	972
			...	...	...	...
1/1/93	5°	0°	1/1/93	5°	0°	972
			1/1/93	0°	0°	975
			1/1/93	10°	0°	977
			...	...	...	...
1/1/93	10°	0°	1/1/93	10°	0°	977
			1/1/93	5°	0°	972
			...	...	...	...
...	...	...	...	...	...	...

Table 9: $SLP\_GROUP(t, lon, lat; SLP\_FLD(t, lon, lat; slp)) \;=\; group_{within\_5^\circ}(SLP)$

Formally, grouping is defined as

$$group_\theta(F) \;\equiv\; \langle\, U_F, S_F, v \,\rangle$$

where

$$\forall p \in U_F,\; v(p) \;=\; \sigma_{\theta,p}(F).$$

4.3.2 Partition *partition*

In addition to grouping, we allow related coordinate records in a field to be collected by partitioning the field into a set of disjoint subfields based on a partition function. The coordinate records in each partition map to the same value in the range of the partition function. Table 10 shows how the sea-level pressure field can be partitioned based the month at which the sea-level pressure value is record. After partitioning, we can then analyze monthly-variation of sea-level pressure.

Given field $F(a_1, \ldots, a_m; b_1, \ldots, b_n)$, and partition function $part : (C_F \times V_F) \rightarrow R$, defined for coordinate records of the field, partition is defined as

$$partition_{part}(F) \;=\; \langle\, S, S, \emptyset \,\rangle$$

SLP_FLD			
t	lon	lat	slp
1/1/93	0°	0°	975
1/1/93	5°	0°	972
1/1/93	10°	0°	977
...	...	...	...
1/2/93	0°	0°	970
1/2/93	5°	0°	982
1/2/93	10°	0°	975
...	...	...	...
2/1/93	0°	0°	992
...	...	...	...
...	...	...	...

Table 10: $SLP\_PART\{\ SLP\_FLD(t, lon, lat; slp)\ \}\ =\ partition_{month(SLP.t)}(SLP)$

where

$$S\ =\ \{\ \sigma_{part(a_1,\ldots,a_m,b_1,\ldots,b_n)=r}(F)\ \mid\ r \in R\ \}.$$

4.4 Sequenced Field Selection

The operators that we have described so far are set-oriented. However, sometimes there is need to manipulate field data that are ordered. We extend the selection operator to allow selection based on temporal(sequential) conditions or event patterns by adopting the basic temporal logic operators, weak next *next*, weak previous *prev*, strong until *until*, and strong since *since*, defined in [34] for reasoning on sequenced data. More complicated predicates can be derived from this basic set of temporal constructs.

Given a field $F \in [C_F, V_F]$, the total element ordering predicate $\prec$: $S_F \times S_F \rightarrow \mathcal{BOOL}$ implicitly defines defines a function $\Phi_\prec : S_F \rightarrow \{\ 1, \ldots, card(S_F)\ \}$ which maps each coordinate in the sampled coverage into its absolute position according to the ordering specified by $\prec$. We denote the point occupying the ith position in the sampled coverage as $\langle S_F, i \rangle$. The expression $\langle S_F, i \rangle\ \models\ P$ means that the ith point in sampled coverage L satisfies the non-temporal predicate P. The semantics of $\models$ is specified in table 11.

Formally, sequential selection is defined as

$$\sigma_{\prec,\models P}(F)\ =\ \langle\ U_F, S, v_F\ \rangle$$

where

$$S\ =\ \{\ p \in S_F\ \mid\ \langle S_F, \Phi_\prec(S_F, p)\rangle\ \models\ P\ \}.$$

$$
\begin{array}{rcl}
\langle S_F, i\rangle \models \neg P & \text{iff} & \langle S_F, i\rangle \not\models P \\
\langle S_F, i\rangle \models P \vee Q & \text{iff} & \langle S_F, i\rangle \models P \vee \langle S_F, i\rangle \models Q \\
\langle S_F, i\rangle \models \textit{next}\ P & \text{iff} & (1 \le i < n) \Rightarrow \langle S_F, i+1\rangle \models P \\
\langle S_F, i\rangle \models \textit{prev}\ P & \text{iff} & (1 < i \le n) \Rightarrow \langle S_F, i-1\rangle \models P \\
\langle S_F, i\rangle \models P\ \textit{until}\ Q & \text{iff} & \exists j : i \le j \le n. \langle S_F, j\rangle \models Q \wedge \forall k : i \le k < n. \langle S_F, k\rangle \models P \\
\langle S_F, i\rangle \models P\ \textit{since}\ Q & \text{iff} & \exists j : 1 \le j \le i. \langle S_F, j\rangle \models Q \wedge \forall k : j < k \le i. \langle S_F, k\rangle \models P
\end{array}
$$

Table 11: Semantics of $\models$

For example, the sea-level pressure minima in each cyclone track in CYC are totally ordered by their time. $Convert^V_{\sigma_{\models\ true\ since\ slp<972}}(CYCS)$ selects the parts of cyclone tracks in which sea-level pressure has previously dropped below 972mb.

5 Implementation Issues

We have described the field model and its algebra in previous sections. In this section, we briefly discuss some query processing and optimization issues related to the implementation of a geoscientific database system based on the field model.

5.1 Query Execution

We have prototyped a parallel dataflow query processor on massively parallel processors (e.g., IBM SP1, Intel Paragon) for queries expressed in the model. This is an important component of *QUEST*, a prototype data analysis environment providing content-based access to massive geoscientific datasets that is under development at UCLA. The implementation of the query processor is based on the Volcano extensible query execution environment[16]. Significant extensions and changes to the environment that we have made to the system include:

- Implementation of field operators in addition to existing relational operators.
- Control of data passing granularity in dataflow environment through the use of nest/unnest operators.
- Replacement of socket-based communication with PVM (Parallel Virtual Machine) [14] for interoperability. PVM enables a network of heterogeneous Unix machines to behave as a single large parallel computer.

- Support of user-defined functions and operators, hence allowing scientists to introduce application-specific operators.
- Extension to the type system to support complex field objects and spatio-temporal datatypes.
- Support the transfer of large, complex data items between processes running on different machines; Volcano message passing protocol only supports the transfer of simple data items that fit in contiguous memory.

5.1.1 Stream/Dataflow Processing

We anticipate several patterns of access to the raw data. Often a user would want to access relatively small amounts of data for which the index ranges are specified. The typical example here is the retrieval of some subarray from a multi-dimensional array stored in some regular fashion. The location of the data in the subarray are usually indexed or can be directly calculated easily. However, more sophisticated query processing techniques are needed to handle more complex queries.

Consider the query to retrieve the average sea-level temperatures in 5° regions centered around points on each cyclone track. The derivation of the field $CYCPT\_AVGTMP$ (Table 8) shows how this query can be expressed in the field algebra. Figure 6 shows a possible query execution plan for the query.

To exploit advantages offered by stream processing techniques for complex queries involving large amount of data, the plan ensures that simple cyclone track point data are sorted to match the natural ordering of more complex and massive temperature data to reduce the need for intermediate data buffering. Node (e) scans slices of useful temperature field data from HDF files in a specified order. In this case, since no selection criteria for temperature has been specified in the query, all temperature data are returned in time order. At the same time, node (i) scans a spatial-temporal data structure for qualified cyclone tracks (in this query, since no restrictions on cyclone tracks is imposed, all cyclones are scanned). Then nodes (h), (g) and (f) decompose the cyclone tracks from node (i) and sort points on cyclone tracks based on time. The join operator (d) can then join temperature fields with related cyclone track points recorded at the same time without having to buffer more than one of them at any time. Finally, nodes (b) and (c) calculates the average sea-level temperature at each point on the trajectory of a cyclone track. Cyclone tracks can then be recreated by nesting before any statistics are calculated on a per-cyclone basis.

In the QUEST query execution engine, we introduced a unique mechanism to control the granularity of data passing by means of the physical nest and unnest operators (the implementations, not the logical operators). We use the variable value associated with a field's outermost coordinate point as the unit of data passing between processes. It adds another dimension to our control of the data as we can vary the message size by changing the nesting of the field to be

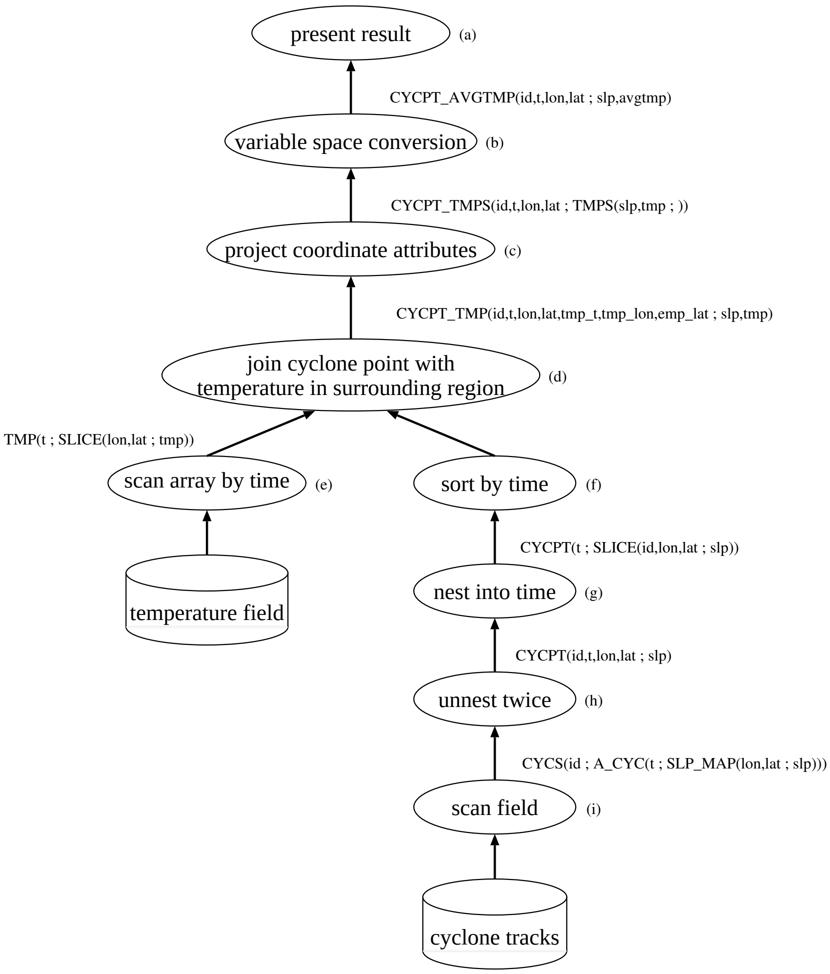

Figure 6: Correlate sea-level pressure with average temperature along cyclone tracks.

passed. Figure 7 shows how sea-level pressure data on cyclone tracks are “chunked” and passed differently during query execution before and after it is nested at node (g).

Contrast the query execution plan with the naive approach of joining temperature fields with complete cyclone tracks covering long periods of time, one at a time. One possible implementation is to discard a temperature field after it is joined with a cyclone track. This implementation is extremely inefficient because cyclone tracks heavily overlap and hence the same temperature field may have to be extracted from disk multiple times any time it is needed, which can be often. An alternative is to keep temperature fields in memory in anticipation of future use. This approach may seem appealing at first because of reduced disk I/O as a result of data caching. However, a significant amount of disk I/O is still needed since the caching scheme degenerates quickly as the limited amount of main memory only allows a small portion of large temperature fields to be cached.

Data analysis applications (e.g., time series analysis, visualization) require large amounts of

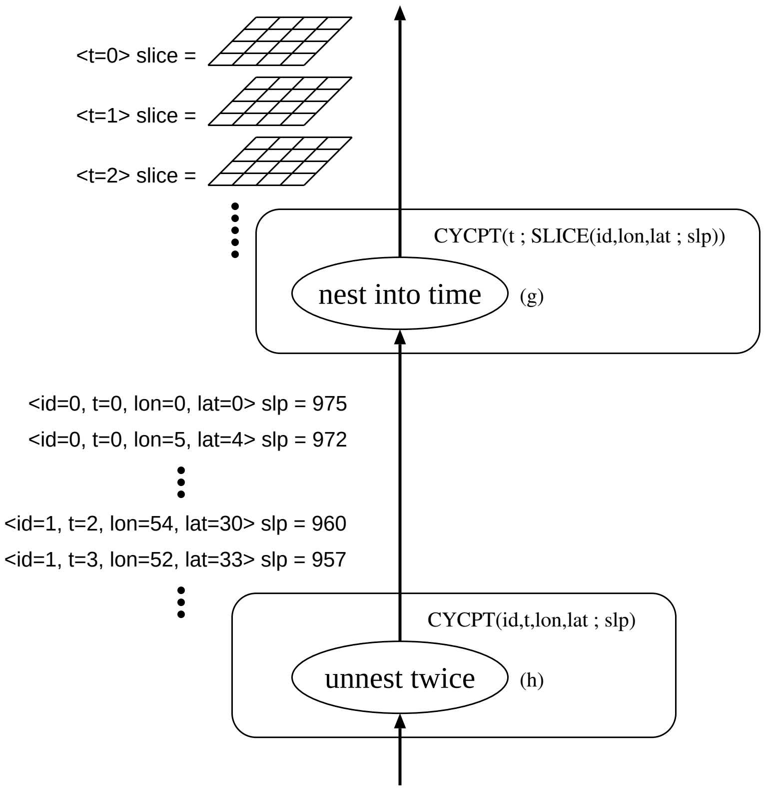

Figure 7: Controlling the granularity of message passing with a field's nesting.

data to be accessed in some order in order to produce the result. The use of stream processing algorithms can help minimize paging of data between main memory and secondary storage and possibly tertiary storage. Stream processing techniques are especially attractive when the physical data organization matches the logical data access pattern, in which case I/O overhead is minimized.

5.1.2 Parallelization

Many different parallelization techniques have been developed for dataflow (or stream) query processing environments [16]. The query execution tree for a complex query usually combines several types of parallelism.

Figure 8a and 8b shows examples of inter-operator parallelism, in which each operator is executed by one of the concurrent processes. Pipeline parallelization is the simplest inter-operator parallelization strategy where processes form a pipeline. Each process obtains data from one "producer" and feeds to one "consumer" after processing. On the other hand, a process involved in bushy parallelization can consume data produced by more than one producer processing, as in the case of the process executing join in Figure 8b. Processes in different

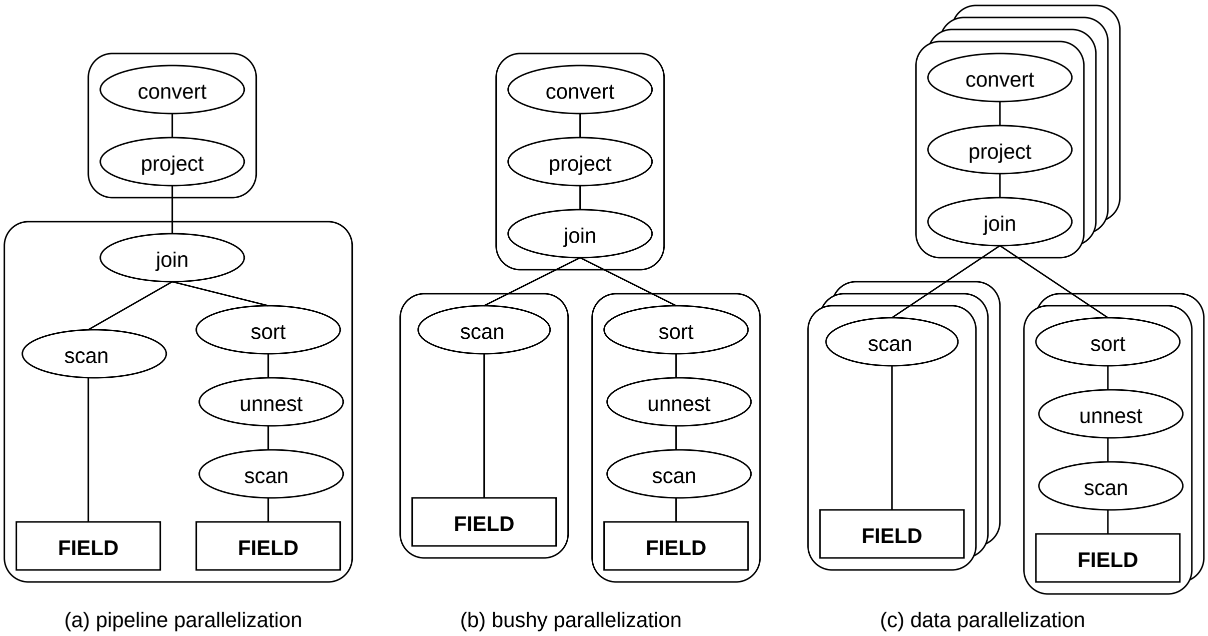

Figure 8: Parallelization of query execution plan.

branches of the query execution tree can be executed independently.

Intra-operator parallelization provides another dimension of parallelism in addition to inter-operator parallelization, It allows an operator to be executed in several processes, with each copy of the operator handling a (disjoint) subset of the input data. For example, in Figure 8c, join is executed in 4 concurrent processes. Their producer processes partition their output data into 4 subsets and feed each of the subset to a process containing a copy of the join. The partitioning can be based, for example, on a hash function on the key of a data item.

5.1.3 Content-Based Indexing

In addition to traditional data indexing techniques, content-based indexes can be extracted from primitive geoscientific field data and can then coexist with the raw data in the system as fields. These extracted features are relatively small in size and can be stored in special purpose DBMSs or data structures to allow efficient processing and access. For example, spatial access methods like quadtree[36] can be used for spatial extents, while spherical point access method (e.g., Quaternary Triangular Mesh[18], Sphere Quadtree [9]) are useful for spherical spaces.

Using content-based indexing techniques, we can generate alternative query execution plans for the query to calculate average sea-level temperature on cyclone tracks. The query may start by selecting a set of cyclone tracks which is used to limit the spatio-temporal range of the sea-level pressure to process. After this step is completed, we join the intermediate result with the sea-level temperature data. In preparation for the join we can process the selected tracks, e.g., we can sort them by starting time and see what time periods are overlapping by at least one cyclone in the set. Only sea-level temperature data corresponding to such time periods need be examined. This approach may allow us to significantly reduce the amount of I/O overhead since we don't have to scan the complete sea-level temperature field.

5.2 Query Optimization

Scientific data are typically large in volume and queries can be complex and computationally expensive. As a result, optimization and parallelization of scientific queries, and the physical organization of scientific data are extremely important inter-dependent problems that have to be studied in order to deliver the kind of performance required by scientists. Although a great deal of work has been done in optimization and parallelization of relational queries, relatively little work has been done studying issues specific to scientific databases. One notable exception can be found in [43].

Extensible query optimization [2, 7, 21, 22] is one of the major issues studied by researcher interested in building extensible database systems. The Volcano optimizer generator [17, 3] provides a flexible and extensible framework for constructing optimizers. We plan to use Volcano as a basis to develop a query optimizer for the field model's algebraic query language. In particular, algebraic query transformation techniques similar to those employed in relational database systems can be used by the field query optimizer to generate alternative query execution plans. Using heuristic search rules and cost estimates, the optimizer searches for the query execution plan with the lowest estimated cost and executes it.

The following are some of the main query optimization issues related to field query optimization that we are studying:

- **Physical organization.** The layout of the data on a disk can make a large difference in the number of I/O operations required. For example, a common operation on a multidimensional array is to retrieve a specified subarray, which efficiency is greatly affected by the way the array is "tiled" in secondary/tertiary storage.

  While we are concerned with this question in isolation, we also believe that optimization of the layout of the data on a storage device is only one of the issues. Taking a more global view, datasets will most often be permanently stored on tertiary storage (e.g. tape jukebox) and parts of the dataset will be staged to optical disk or magnetic disk when required by an application. The movement of the data to the disks offers an opportunity to organize the (portion of) the dataset retrieved in order to optimize performance for the particular application. For this reason it would be advantageous to be able to reason about the access pattern of the application on this dataset. This suggests a "smart compiler" that can analyze the application code and (a) suggest the most efficient way of organizing the data on disk and (b) optimizing the computation including the disk accesses. Cooperation with other mechanisms such as the disk cache management can also reap performance benefits. The ability to understand what the application "is doing" enhances the ability of the compiler to perform the above mentioned optimizations.

- **Regularity.** A taxonomy of spatial grids is presented in [41]. Figure 9 shows some 2-D grids with different regularities. Cells in a regular and rectilinear grids are aligned with the axes, while distances between points along an axis are arbitrary for a rectilinear grid. A structured grid is logically a regular grid subjected to a non-linear transformation. For geoscientific datasets, sampled coverages are mostly regular, rectilinear, or unstructured. For example, the data created as output of an atmospheric model simulation are usually regular, while in-situ observation data are usually unstructured and randomly scattered. One interesting problem we have encountered concerns the handling of observation data from non-geostationary satellites. Such data are "approximately" structured in the sense that the footprint of a satellite at a particular time can be calculated to within a certain degree of accuracy. We are investigating ways to take advantage of this special property of satellite data during query processing.

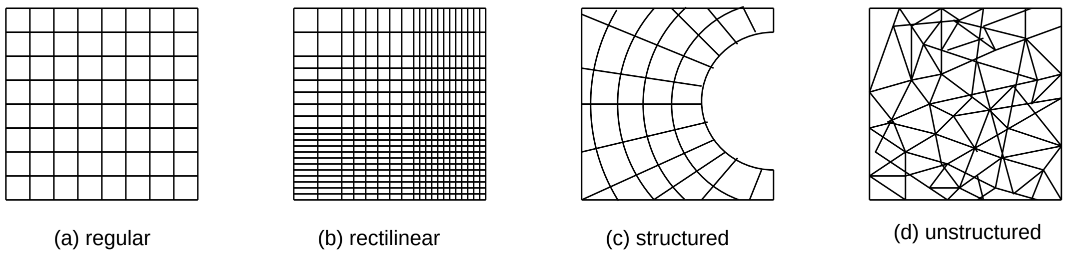

Figure 9: Taxonomy of grid regularity

- **Element Ordering.** Typically, points in a coordinate space have a natural ordering. For example, time is totally ordered, while longitude and months are cyclic. Sequenced geoscientific data are extremely common. However, note that the appropriate element ordering of a field sometimes depends on the application. For example, elements in a 2-dimensional array might be in either column-major order or row-major order. It is also possible that the coordinates are unordered.

  Many geoscientific data analysis applications involve studying the change of some feature or parameter along certain dimensions. For example, cyclone trajectories can be extracted from a sea-level pressure field by tracking the movement of local sea-level pressure minima in time. In a stream query processing environment, a field's coordinate ordering can be used to help determine the field's physical layout so that I/O overhead can be minimized.

- **Interpolation.** Little research has been done on the optimization of queries involving implicit data derived by an interpolation function [31]. For many field operators, as a field is being operating on, its interpolation function increases in complexity. Lets consider the selection operator. After selection, a field can be left with a sampled coverage which doesn't support the interpolation function. For example, it is possible that none of the points that satisfy the selection condition lies in the sampled coverage. In this case, the sampled coverage of the result field is empty and hence we cannot derive variable values based on values associated with points in the sampled coverage. It implies that both the interpolation function and the data from the input field have to be carried around with the result field if derivation of variable values (resampling) is needed at a later stage.

In the field model, resampling and relational operators are transitive. As a result, query execution is often more efficient if resampling is pushed as far down the execution tree and performed as early as possible. An exception would be when early resampling vastly increases the amount of data that are explicitly stored (i.e., sample coordinates), causing a higher overhead in data communication between nodes in the query execution plan..

- **User-defined functions.** The field model allows the introduction of user-defined function and predicates [42] to perform complex application-specific operation. It has been shown [24] that the heuristic used by most query optimizers to push selection actually produces query plans with worse performance if the selection involves expensive foreign predicates. As a result, it is necessary to develop a scheme to capture the various aspects of a foreign function (e.g., startup cost [13], semantics [8]) in order for it to be properly handled by the query optimizer.

6 Conclusions

In this paper, we have described a flexible data model capturing the unique properties of geoscientific data. The field model identifies the different roles of coordinates and variable values in geoscientific dataset, hence allow the structure (e.g., regularity, ordering) of coordinates to be be taken advantage of for efficient processing and storage. Moreover, the fact that geoscientific datasets are large in size, and often sparse dictates the need for support of implicitly stored data which are derived by interpolation functions.

Based on the field model, we have prototyped a parallel dataflow query processor on massively parallel processors as part of the *QUEST* geoscientific data analysis environment that is currently under development at UCLA. Early experiences with the system are promising. The field model is flexible enough to express different geoscientific data types and the query execution engine (with minimal query optimization) delivers performance comparable to hand-tailored programs for many queries.

We have identified important geoscientific query optimization issues on which we are concentrating our current research effort. Such overlooked problems include the use of interpolation functions and expensive foreign user-defined functions in a query.

References

[1] Advanced Visual Systems, Inc., *AVS User's Guide*, May 1992.

[2] L. Becker, and R.H. Guting, "Rule-Based Optimization and Query Processing in an Extensible Geometric Database System," *ACM Transaction on Database Systems*, vol. 17, no. 2, 1992.

[3] J.A. Blakeley, W.J. McKenna, and G. Graefe, "Experiences Building the Open OODB Query Optimizer," *Proceedings of ACM SIGMOD Conference*, 1993.

[4] D.M. Butler, and M.H. Pendley, "A Visualization Model Based on the Mathematics of Fiber Bundle," *Computers in Physics*, Sep/Oct 1989.

[5] D.M. Butler, and S. Bryson, "Vector-bundle Classes Form Powerful Tool for Scientific Visualization," *Computers in Physics*, Nov/Dec 1992.

[6] D.M. Butler, and C. Hansen, "Visualization '91 Workshop Report: Scientific Visualization Environments," *Computer Graphics*, Vol. 26, No. 3, Aug 1992.

[7] M.J. Carey, D.J. DeWitt, G. Graefe, D.M. Haight, J.E. Richardson, D.T. Schuh, E.J. Shekita, and S. Vandenberg, "The EXODUS Extensible DBMS Project: An Overview," *Readings on Object-Oriented Database Systems*, D. Maier and S. Zdonik (eds.), Morgan Kaufman, San Mateo, CA, 1990.

[8] S. Chaudhuri, and K. Shim, "Query Optimization in the Presence of Foreign Functions," *Proceedings of 19th VLDB Conference*, Dublin, Ireland, 1993.

[9] G. Fekete, "Sphere Quadtrees: a New Data Structure to Support the Visualization of Spherically Distributed Data," *Proceedings of SPIE/SPSE Symposium on Electronic Imaging Science and Technology*, Feb 1990.

[10] J.C. French, A.K. Jones, and J.L. Pfaltz, "Scientific Data Management," *ACM SIGMOD Record*, Vol. 19, No. 4, Dec 1990.

[11] J. C. French, "Support for Scientific Database Management," *Statistical and Scientific Databases*, Z. Michalewicz (ed.), Ellis Horwood, New York, NY, 1991.

[12] Gallop, J., "Survey of data models for scientific visualization systems", *Computer Graphics. Computer Animation, Virtual Reality, Visualization*, 1991.

[13] R. Gamboa, D. Chimenti, and R. Krishnamurthy, "Towards an Open Architecture for LDL," *Proceedings of 15th VLDB Conference*, 1989.

[14] G.A. Geist, and V.S. Sunderam, "Experiences with Network-Based Concurrent Computing on the PVM System," *Concurrency: Practice and Experience*, Vol. 4, No. 4, Jun 1992.

[15] S. Ginsburg, and X. Wang, "Pattern Matching by Rs-Operators: Towards a Unified Approach to Querying Sequenced Data," *Proceedings of ACM Symposium on Principles of Database Systems*, 1992.

[16] G. Graefe, and D.L. Davison, "Encapsulation of Parallelism and Architecture-Independence in Extensible Database Query Execution," *IEEE Transactions on Software Engineering*, Vol. 19, No. 8, Aug 1993.

[17] G. Graefe, and W.J. McKenna, "The Volcano Optimizer Generator: Extensibility and Efficient Search," *Proceedings of IEEE Conference on Data Engineering*, Vienna, Austria, Apr 1993.

[18] M. Goodchild, and S. Yang, "A Hierarchical Spatial Data Structure for Global Geographic Information Systems," Technical Paper 89-5, National Center for Geographic Information and Analysis, University of California, Santa Barbara, 1989.

[19] R.H. Guting, and R. Zicari, "An Introduction to the Nested Sequences of Tuples Data model and Algebra," *Nested Relations and Complex Objects inb Databases*, S. Abiteboul, P.C. Fisher, H.J. Schek (eds.), Springer-Verlag, 1989.

[20] R.H. Guting, R. Zicari, and D. Choy, "An Algebra for Structured Office Documents," *ACM Trans. Information Systems*, vol. 7, 1989.

[21] L. Haas, J.C. Freytag, G. Lohman, and H. Pirahesh, "Extensible Query Processing in Starburst," *Proceedings of ACM SIGMOD Conference*, Portland, OR, May 1989.

[22] L. Haas, et al., "Starburst Mid-Flight: As the Dust Clears," *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, no. 1, 1990.

[23] R.B. Haber, B. Lucas, and N. Collins, "A Data Model for Scientific Visualization with Provisions for Regular and Irregular Grids," *Proceedings of Visualization '91 Conference*, San Diego, CA, Oct 1991.

[24] J. Hellerstein, and M. Stonebraker, "Predicate Migration: Optimizing Queries with Expensive Predicates," *Proceedings of ACM SIGMOD Conference*, 1993.

[25] IBM Corporation, *IBM Visualization Data Explorer: User's Guide*, 1992.

[26] The Khoros Group, University of New Mexico, *Khoros User's Manual*, 1992.

[27] W. Kim, "Object-Oriented Approach to Managing Statistical and Scientific Databases," *Proceedings of 5th International Conference on Statistical and Scientific Database Management*, Charlotte, NC, Apr 1990.

[28] P. Kochevar, Z. Ahmed, J. Shade, and C. Sharp, "A Visualization Architecture for the Sequoia 2000 Project," Sequoia 2000 Technical Report 93/35, Sep 1993.

[29] A. Makinouchi, "A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model," *Proceedings of VLDB Conference* 1977.

[30] C.R. Mechoso, C.C. Ma, J.D. Farrara, and J.A. Spahr, "Simulations of Interannual Variability with a Coupled Atmosphere-Ocean General Circulation Model," *Proceedings of 5th Conference on Climate Variations*, American Meteorology Society, Boston, MA, 1991.

[31] L. Neugebauer, "Optimization and Evaluation of Database Queries Including Embedded Interpolation Procedures," *Proceedings of ACM SIGMOD Conference*, 1991.

[32] P. Pister, and R. Traunmuller, "A Database base Language for Sets, Lists, and tables," *Information Systems*, Vol. 11, No. 4, Dec 1986.

[33] Research Systems, Inc., *IDL User's Guide*, Jan 1993.

[34] J. Richardson, "Supporting Lists in a Data Model (A Timely Approach)," *Proceedings of 18th VLDB Conference*, Vancouver, British Columbia, Canada, 1992.

[35] M.A. Roth, H.F. Korth, and A. Silberschatz, "Extended Algebra and Calculus for Nested Relational Databases," *ACM Transactions on Database Systems*, Vol. 13, No. 4, Dec 1988.

[36] H. Samet, *Applications of Spatial Data Structures*, Addison-Wesley, Reading, MA, 1990.

[37] H.J. Schek, and M.H. Scholl, "The Relational Model with Relation-Valued Attribute", *Information Systems*, Vol. 11, No. 2, 1986.

[38] A. Segev, and A. Shoshani, "Logical Modeling of Temporal Data," *Proceedings of ACM SIGMOD Conference*, San Francisco, CA, May 1987.

[39] A. Shoshani, and H.K.T. Wong, "Statistical and Scientific Database Issues," *IEEE Transactions on Software Engineering*, Vol. 11, No. 10, Oct 1985.

[40] A. Shoshani, "Properties of Statistical and Scientific Databases," *Statistical and Scientific Databases*, Z. Michalewicz (ed.), Ellis Horwood, New York, NY, 1991.

[41] D. Speray, and S. Kennon, "Volume Probes: Interactive Data Exploration on Arbitrary Grids," *Computer Graphics*, Vol. 24, No. 5, Nov 1990.

[42] M. Stonebraker, J. Anton, and E. Henson, "Extending a Data Base System with Procedures," *ACM Transactions on Database Systems*, Vol. 12, No. 3, Sep 1987.

[43] R. H. Wolniewicz, and G. Graefe, "Algebraic Optimization of Computations over Scientific Databases," *Proceedings of 19th VLDB Conference*, Dublin, Ireland, 1993.