The Sheaf Data Model

A rigorous data model for scientific computing

David M. Butler
Limit Point Systems, Inc.
Livermore, CA 94550
d.m.butler@limitpoint.com

***Abstract*—We summarize the features of the sheaf data model, a rigorous data model for scientific computing.**

Keywords- database management, data models, scientific computing, high performance computing

I. THE NOTION OF A DATA MODEL

In this paper we introduce the sheaf data model, a rigorous data model for scientific computing. We use the term "data model" in the sense originally defined by Codd [3], that is, a data model consists of a class of mathematical objects, the operations on the objects, and a set of constraints on valid instances of the objects. The best known such model is the relational data model.

II. RELATIONAL DATA MODEL

The objects in the relational model are relations on sets, the operations are relational algebra, and the constraints are a fairly complex set of requirements often referred to as "normalization". The relational data model revolutionized business data processing in numerous ways. The mathematics of the model raised the level of abstraction of applications developed on top of it. Storage specific procedural queries were replaced by declarative SQL queries. Application independent tools such as report generators replaced untold quantities of custom programming. The relational database management system became the main tool for integration and interoperation of diverse sources and formats.

In spite of its success in the business data processing world, the relational model has never been widely used in scientific computing. Succinctly put, the model has two shortcomings for scientific applications: (1) it does not match the way we want to use the data, the table-oriented operations of the model are too low level, and (2) it does not match the way we want to store the data, the normalization conditions conflict with efficient storage and access.

III. DATA MODELS FOR SCIENTIFIC COMPUTING

Over the last several decades there have been numerous attempts to find a data model for scientific computing that would play a role similar to that played by the relational model in business data processing. One approach to such a model is to leverage a unique feature of the scientific domain: unlike business, the scientific domain itself is already mathematized. We don't need to search for the right abstractions, science has spent centuries developing them. We just need to use them.

The fiber bundle model [1] followed this approach. The objects in the model, taken directly from mathematical physics, are sections of fiber bundles over topological spaces while the operations are section algebra and calculus. Informally, a section of a fiber bundle represents some property as a function of position in some physical object and the operations support familiar notions of addition, subtraction, differentiation, etc.. The mathematical machinery associated with fiber bundles and topological spaces provides the generality to deal with a very wide variety of applications.

The fiber bundle model has been used successfully as the basis for a number of systems, the best known being the IBM Data Explorer product, now the OpenDX open source project [5], and the more recent F5 library [4]. However, extensive experience with the fiber bundle model in a variety of settings has shown that while the model does a good job of addressing how we want to use the data, it does not address how to store the data. Storage fundamentally requires a description of how a high level, semantic object is decomposed into a collection of low level primitives. The abstractions of the fiber bundle model simply don't address such decompositions. As a result, implementation of the model requires ad-hoc extensions and these extensions inevitably limit the scope of the implementation.

IV. THE SHEAF DATA MODEL

The sheaf data model addresses decomposition at a fundamental mathematical level. The objects in the model are sheaves of lattice-ordered relations over lattice-ordered relations. The mathematical details of the model are described in [2] but are beyond the scope of this presentation. Informally, finite distributive lattices ("FDL"s) and sheaves provide a rigorous formalism for describing, respectively, parts and the association of attributes with those parts.

The sheaf data model, like the relational model, can be described with a table metaphor, as shown in Fig. 1. As with the relational model, a database is a collection of tables, each table represents a type, each row represents an instance

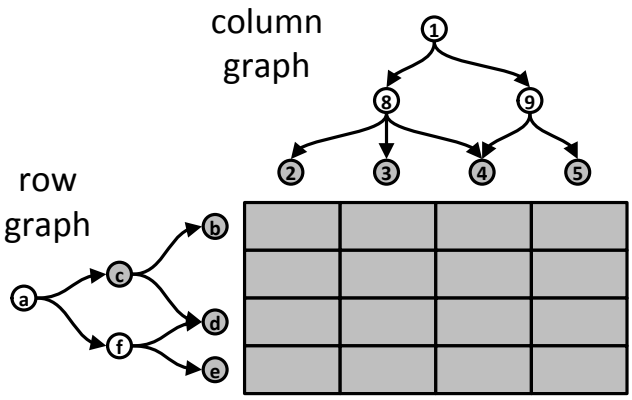

Figure 1. Table metaphor for the sheaf data model.

of the type, and the columns represent attributes of the type.

The rows in the relational model are unordered, but in the sheaf data model the rows are equipped with a client-defined lattice order. An FDL can be thought of as representing the set of all distinct composite parts that can be built from a given set of basic parts, along with the inclusion relationships between them. This information can be represented visually by a directed acyclic graph with two kinds of nodes, one for basic parts and one for composite parts, with the links representing "directly includes" (i,e, the cover relation). The table metaphor for the sheaf model thus has a graph associated with the rows, with a basic part for each row in the table, as shown in the figure, where the basic parts are represented by filled circles. Similarly, the collection of columns, the schema of the table, is also lattice ordered, but the lattice order for the schema is specified by the row order of some other table in the database. Thus schema are first class objects and each table must have another table as its schema. This schema recursion terminates in a special table, the primitives schema table, which uses its own row order as its schema.

The introduction of schema as first class objects sets up a dependent type system that can be used to rigorously represent all the important types of scientific computing. In particular, tabular types, object-oriented types, spatial types, and field types can all be represented within the same unified formalism. We discuss each of these types in the following.

A. *Tabular types*

Both the rows and the columns are considered unordered in conventional relational tables. This corresponds the lattice order being that of a Boolean lattice, represented graphically by basic parts without any links. The relational model is thus contained within the sheaf model as a limiting case.

B. *Object-oriented types*

For object-oriented types, the column order represents the subobject inclusion order associated with single or multiple inheritance (base classes) and aggregation (data members).

Object-oriented types are important in scientific computing because physical properties are represented by a collection of mathematical types (scalars, vectors, tensors, etc.) with a natural object-oriented structure. The sheaf model supports representation of these properties as strongly-typed objects rather than collections of attributes and the schema provides a rigorous connection between the data and its abstract mathematical specification. The fact that the schema for these types are first class objects provides a mechanism for diverse applications to rigorously share such specifications.

C. *Spatial types*

Spatial types, and their discrete representation as meshes, are at the heart of scientific computing. A mesh can be considered a decomposition of a given spatial domain into a set of basic parts, namely the mesh primitives such as zones, faces, edges, or vertices. Every such decomposition, regardless of the specific primitives used, can be represented as an FDL [6]. The nodes in the graph represent parts of space and the links represent spatial inclusion. The FDL provides a rigorous connection from the concrete numerical representation through the theory of cellular complexes to the abstract semantics of the underlying topological space.

The FDL for a mesh is an algebraic representation that supports a broad set of spatial operations such as union, intersection, sum, difference, Cartesian product, and tensor product.

An important feature of the FDL representation is it can represent multiple, concurrent decompositions in addition to the mesh: domain decomposition for parallel processing, parts decomposition for design, decomposition based on the value of some property, geographical or political region decomposition, multiple mesh refinements, and/or any ad hoc client-defined decomposition. Any combination of decompositions can be created at execution time using the lattice operations and without being "designed-in" to the software.

D. *Field types*

The sheaf data model introduces a rigorous notion of schema for field types. Informally, a field is a map from some spatial type to some property type. The schema for a field is given by an algebraic construction involving the lattice describing the rows of the spatial type and the lattice describing the columns of the property type. A wide range of constructions is possible, but in practice the field schema is typically the lattice tensor product of the spatial lattice and the property schema lattice. The field schema concretely describes the field data and provides a rigorous, seamless bridge from the numerical representation to the abstract mathematical specification and semantics of the field. The field schema supports restriction of the field to any part of the domain or any part of the property, an important feature for domain decomposition, parallel processing, visualization, and other applications.

V. FIBER BUNDLE MODEL REVISITED

The objects of the fiber bundle model, sections of fiber bundles over topological spaces, have a natural description within the sheaf data model. Hence, the fiber bundle model

can be implemented on top of the sheaf model. The resulting layered abstractions provide the utility of the fiber bundle model combined with a complete and detailed description of storage that addresses the entire scope of scientific data management.

VI. Applications

The sheaf data model is intended to play a role in scientific data management similar to that of the relational data model in business data management, providing a higher level of abstraction, enabling more general tools and applications, and increasing productivity. Some examples include:

A. *Interoperation of diverse field formats*

Tools to address many of the mesh related "data munging" chores associated with interoperating geometry and property fields defined on different mesh types can be specified and implemented entirely within the sheaf abstractions, without reference to the specific mesh format. In particular, we have implemented a universal "property pusher" tool for moving property fields between meshes, with optional, policy-controlled refinement of the target mesh to meet accuracy requirements. Interoperating N mesh types using such mesh-independent tools requires only O(N) sheaf adapters, not $O(N^2)$ format translators, a major productivity gain.

B. *Visualization of diverse field formats*

The sheaf objects and operations support the implementation of format independent visualization algorithms in a manner analogous to that just described for data translation. To support M visualization techniques for N formats, we need to develop only O(M) format independent visualization routines, not O(N*M) format specific visualization routines.

C. *Effective use of parallel computing*

The sheaf concepts facilitate identification and description of the parallelism inherent in a problem domain at the conceptual level and provide for the propagation and refinement of that information into the computational and persistent data aspects. The concepts and their embodiment as software facilitate platform independent parallel implementations of the data manipulation and visualization algorithms described above and potentially enable implementation of application and platform independent, automatic problem decomposition and distribution services.

D. *Long-term, archival storage*

The model provides a mathematically precise, and provably general method for schematizing technical data. Since the schema captures the mathematical semantics of the data, it does not depend on persistence of the application that generated the data. Data storage using such schema can be implemented in a variety of ways that can evolve with the technology environment.

E. *Formal language and/or ontology*

The sheaf concepts can be developed into a formal data definition and manipulation language. Such a "SheafQL" could play a role similar to the role SQL has played in increasing programmer productivity and facilitating application integration in business information systems. Similarly, the sheaf concepts can potentially be developed into a formal ontology for semantic web applications. Both the formal language and the ontology would inherit the natural expression of parallelism provided by the sheaf concepts.

VII. Current status

We have implemented the sheaf data model as a C++ class library with bindings for Java and Python, and are currently developing a C# binding. The library contains an API for the general sheaf data model and an API for the fiber bundle data model, implemented on top of the sheaf API. Both have been extensively optimized to give asymptotic space and time performance similar to conventional representations. We have proven the efficacy of the model in numerous applications, especially within the oil exploration and production domain. We are currently preparing an open source release.

VIII. Future Work

There are numerous interesting research questions suggested by the model. First and foremost: what is the query language? Others include: what is the transaction model, what storage models are appropriate, and what new applications and tools does the model enable? We invite others to join us in pursuing these questions.

Acknowledgement

Original research and development of the sheaf data model was funded by subcontracts B347785, B515090, and B560973 of prime contract W-7405-ENG-48 with the Department of Energy National Nuclear Security Administration (DOE/NNSA). Ongoing development has been funded by Shell GameChanger.

References

[1] Butler, D.M. and Pendley, M.H. 1989. A visualization model based on the mathematics of fiber bundles, Computers in Physics. 3, 5 (1989), 45-51.

[2] Butler, D. M. 2012 The Sheaf Data Model. Technical Report, Limit Point Systems, Inc., http://www.limitpoint.com/images/Publications/The%20Sheaf%20Data%20Model.pdf.

[3] E. F. Codd. 1970. A relational model of data for large shared data banks. Commun. ACM 13, 6 (June 1970), 377-387. DOI=10.1145/362384.362685 http://doi.acm.org/10.1145/362384.362685

[4] Fiber Bundle HDF5 Library, http://sciviz.cct.lsu.edu/projects/F5

[5] OpenDX, www.opendx.org

[6] Shapiro, V 1997. Maintenance of geometric representations through space decompositions. Int. J. of Comp. Geometry & Applications, 7, 1 & 2 (1997) 21-56.