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

What is an Astronomical Data Model?

A. Farris and R. J. Allen

Space Telescope Science Institute, 3700 San Martin Dr., Baltimore, MD 21218

Abstract. Using techniques from semantic data modeling and object-oriented data modeling, an astronomical data model is identified as a kind of conceptual analysis. The general characteristics of such a model are presented, as well as its relationship to computer-based data structures and the FITS standard for transporting data. The effective use of data models in relation to instrument specific data, in conjunction with data archives, and in fundamental areas of the data analysis is discussed.

1. Origins of the Concept of a Data Model

The basic concept of a data model originated within the database community as the logical model of an integrated database management system (DBMS). The early DBMSs employed hierarchical, network, and relational data models that were directly implemented by the DBMS software. The entity-relationship model was introduced as a generalization of these previous models and is a more intuitive approach to modeling data relationships. It is widely used as a conceptual tool for relational database design (Batini 1992). Semantic data modeling is an outgrowth of the entity-relationship model in which additional structures are employed to portray more complex data relationships. Object-oriented modeling techniques (Booch 1994) are closely related to semantic data modeling, the major difference being that the object-oriented techniques add behavior to characterize entities rather than being restricted to data attributes.

A semantic data model or an object-oriented data model portrays named objects and relationships between objects. Each object has a set of specific attributes characterized by data items of a specific type. The relationships between objects also have names and quantitative aspects, one to one, one to many, many to many, etc. Relationships may have data attributes as well. For example, if Book and Student are objects in a library data model, then Borrows represents a relationship between Book and Student and the data item, due-date, belongs to the Borrows relationship. Most relationships tend to be binary, but they can be n-ary as well. For example, Meets may be a relationship between three objects, a Course, a Room, and a Time ("a course meets in a room at a time"). There are two other important ways in which objects may be related. One object may be a part of another object (an Engine is part of a Car) and one object may be a specialization of another object (a Rectangle is a kind of Polygon).

There are a number of graphical techniques for representing both semantic data models and object-oriented models. These techniques include representations for all of the basic features used in modeling objects and their relationships, including whole-part and generalization-specialization relationships. They have the advantage of being easily understood and, yet, can model data complexity to arbitrary levels of detail. They are also independent of any specific computer language, enabling them to be implemented in many software environments.

2. Characteristics of an Astronomical Data Model

A data model, as depicted above, is a representation of some state of affairs that exists within the domain of the problem under consideration. It is the result of a conceptual analysis of that problem domain. An astronomical data model is the result of a conceptual analysis of the characteristics of astronomical data and the relationships that obtain between those kinds of data. This analysis is mapped onto a set of graphical or linguistic conventions, able to faithfully represent the characteristics and complexity of the data. Each component of the mapping must have a physical interpretation. The entire model designates a state of affairs that exists, has existed, or might possibly exist in reality.

There has been a tendency within astronomy to confuse computer-based data structures with a data model. Data structures are used to implement a data model. In themselves, apart from any physical interpretation, they are merely abstract data structures. The physical interpretation is an essential part of the concept of an astronomical data model. It is in virtue of this aspect that a data model can be said to be true or false. In other words, data models have meaning; they make assertions about the nature of reality.

It should be clear, at this point, why FITS (NOST 1993) is not an astronomical data model. In its current form as a standard transport mechanism, FITS does not require a physical interpretation. All astronomical keywords and even units are optional. FITS is merely a convention for exchanging bits in a manner that is independent of hardware. A FITS image might have nothing to do with astronomy; it might be a bit mapped image of Greek text. However, the FITS standardization process can become a vehicle for defining standard models of the basic astronomical data concepts. To be a data model of an astronomical image, a FITS image must *require* sufficient astronomical keywords to provide a meaningful interpretation, including units and a world coordinate system.

3. Uses of Data Models within Astronomy

In considering astronomical data handling there are three specific areas where data models can make an effective contribution: in characterizing instrument specific data, in providing a higher conceptual view for searching data archives, and as the basis for implementing a more open approach to the data analysis process.

One of the most vexing areas of modern astronomy, for both users and software developers, is in dealing with modern astronomical instrumentation. Whether it is optical, radio, or X-ray astronomy, instrumentation is complex, with many different modes of operation, each having its own capabilities, calibration procedures, and data formats. Moreover, newer instrumentation will become more complex as basic hardware shrinks in size and increases in functionality. The use of object-oriented modeling techniques can greatly aid in coping with this increased complexity. Graphical representations of data relationships, corresponding to modes of operation of astronomical instrumentation, can be used as an effective means of communication between instrument scientists developing the hardware, end-users attempting to understand how to use the instrument, and software developers responsible for developing the calibration and data analysis procedures. Since such representations are independent of specific computer languages and software development environments, they can serve to capture fundamental design features in an implementation independent manner and in a manner that makes data relationships intelligible to a broader spectrum of people.

Large archives of astronomical data already exist and will become increasingly significant for research in astronomy. The current archives are difficult to use, with no standardization in interface software. While most archives return data in the form of FITS files, there is no standardized way of presenting the data within the FITS file. In short, there is no standard data model. This situation is exacerbated by the fact that these archives, in many cases, store data internally in small, independent FITS files with little or no way of dealing with collections of files or forming relationships between those files. This latter point is particularly acute in storing calibration data, which is usually tied to specific instruments. As archives grow in size and complexity, better methods must be found to store and access data within these archives.

One approach to dealing with this data complexity is illustrated by the Space Telescope Data Archive and Distribution Service (ST-DADS) (Schreier 1991). Scientific and engineering data from the *Hubble Space Telescope* are archived on WORM optical disks in the form of self contained FITS data sets. The data in the FITS keywords is automatically captured and used to populate a relational database that forms a catalog describing the contents of the data archive. This relational database is used to find and request that data be placed on-line for access. The complexity of the *Hubble Space Telescope* data is reflected in the complexity of the relational database that is the catalog portion of ST-DADS. This database has over 1500 attributes distributed among more than 40 relational tables. Using SQL, the standard relational database interface, for a database of this complexity is a challenging experience. Writing a correct SQL query to satisfy even a simple request, i.e., simple from an astronomer's point of view, is a feat that few can master.

To cope with this complex catalog, an object-oriented user interface to ST-DADS, called StarView (Williams 1993), was developed. At its heart is an automated query generator that is based on a higher-level view of the database using the extended entity-relationship model. This query generator uses a system called QUICK (Semmel 1993) which creates contexts for generating SQL statements based on the conceptual model. Typical queries generated vary in size from a few lines to as much as seven pages of SQL statements. A example of a query from an astronomer might be: "find dataset name, archive class, and date of archives containing information about pulsars in a specified region of the sky." In this particular case, because of the large number of tables in the database, the generated SQL statement is over one page of text. Without the automated query generator, it required a designer who is intimately familiar with the database more than twenty minutes to construct a rough draft of the same query and this did not include the time to debug it. This approach is a good illustration of using high-level models as vehicles to cope with underlying complexity.

As instrumentation becomes more complex, the calibration and data analysis process becomes correspondingly more complex. It is not an easy matter to discover what users want in an "ideal" data analysis system. (It is much easier to find out what they do not like.) However, users appear to want the following general features: (1) They want the easy tasks done in a straightforward and intuitive manner. (2) They want considerable flexibility in doing the difficult tasks. (3) They want the entire data analysis scheme wrapped in an intuitive graphical user interface. (4) They want to change what they do not like. (5) They want to be able to add their own custom developed software tasks to the analysis scheme without being told what computer language to program in or what packages they can or cannot use.

One approach to providing such a scheme is to view the system as consisting of a loosely coupled set of independent tasks. Such a system could be implemented provided there is a common data model recognized by all the tasks. If such a data model were defined within a given analysis environment, it would serve as the mechanism that unifies the set of independent tasks. New tasks could be added, provided they implemented the same data model. Such an approach would result in a much more open analysis environment than exists at present. A first step in such a process would consist of defining standard models of basic astronomical concepts, such as "image" and "spectrum," that are independent of particular computer languages and particular data analysis systems.

References

Batini, C., Ceri, S., & Navathe, S. 1992, Conceptual Database Design, An Entity-Relationship Approach (New York, Benjamin/Cummings)

Booch, G. 1994, Object-Oriented Analysis and Design with Applications, second edition (New York, Benjamin/Cummings)

NASA Office of Standards and Technology 1993, Definition of the Flexible Image Transport System (FITS) (Greenbelt, NASA/OSSA)

Schreier, E., Benvenuti, P., & Pasian, F. 1991, in Databases & On-Line Data in Astronomy, eds. M. Albrecht & D. Egret, (Dordrecht, Kluwer), p. 47

Semmel, R., & Silberberg, D. 1993, Telematics and Informatics, 10, 301

Williams, J. 1993, in Astronomical Data Analysis Software and Systems II, ASP Conf. Ser., Vol. 52, eds. R. J. Hanisch, R. J. V. Brissenden, & J. Barnes (San Francisco, ASP), p. 100