COMPUTERS IN ENGINEERING 1994

Volume 2

Finite Element Analysis
CAD/CAM
Fluid Mechanics and Energy Systems
AI/Feature-Based Design and Manufacturing
Computers in Engineering Education
Robotics and Controls
Multimedia Application/Interface

94-55

Computers in Engineering – 1994
Volume 2
ASME 1994

STORAGE OF SPATIAL DATA IN SEMANTIC DATABASES

Naphtali Rishe
School of Computer Science
Florida International University
Miami, Florida

Qiang Li
Dept, of Computer Engineering
Santa Clara University
Santa Clara, California

ABSTRACT. This paper describes how the semantic binary database model [Rishe-92-DDS] and its implementation [Rishe-91-FS] are extended to represent spatial data: functions over multi-dimensional space, e.g. the ocean temperature as a function of the four-dimensional space-time. The implementation, utilizing extended linear quad-trees, is compact in storage and allows efficient resolution of queries.

Semantic data models

The central notion of semantic models is the concept of *object*, which is any real world entity that we wish to store information about in the database. The objects are categorized into classes according to their common properties. These classes, called *categories*, need not be disjoint — that is, one object may belong to several of them. Further, an arbitrary structure of subcategories and supercategories can be defined. The representation of the objects in the computer is invisible to the user, who perceives the objects as real-world entities, whether tangible, such as persons or cars, or intangible, such as observations, meetings, or desires. The database is perceived by its user as a set of facts about objects. These facts are of three types: facts stating that an object belongs to a category; facts stating that there is a relationship between objects; and facts relating objects to data, such as numbers, texts, dates, images, tabulated or analytical functions, etc. The relationships can be of arbitrary kinds; for example, stating that there is a many-to-many relation *address* between the category of persons and texts means that one person may have an address, several addresses, or no address at all. [Rishe-92-DDS]

The relational database model, proposed by E. Codd in 1971, has become the state of the art of commercial database management. This model has, in a mathematically elegant way, presented the database user with an abstraction of data, isolating the user from the physical representation of data in computer storage. The data is presented to the user as a collection of tables. Each table is a set of rows. There are two types of tables: tables representing the application's objects (object tables) and tables representing relationships between objects (relationship tables). The object tables roughly correspond to categories in the semantic models. Each object table consists of one or more columns (jointly called the *key*)

This work is sponsored in part by grants from US DOD/BMDO & ARO, NATO, and Enterprise Florida.

that identify the objects, and several other columns that display data about the objects such as numbers or character strings. Every object must have a unique key value, such as a social security number for a person, or a street name plus a house number for a home. The key must be known at all times and may never change (or the database will be corrupted). Along with the key, an object table's row contains data about the object, such as the person's address, and some relationships to other objects. The latter relationships are represented by the keys of the related objects — for example, the social security number of the person's spouse. This does not work for many-to-many relations, for which the other type of tables must be used: the relationships tables. As far as the system is concerned, the sets of objects of different tables are disjoint (although, the user can de-facto link between rows of different tables by using identical key values, but this causes immense problems in updating and querying the database). The relationships tables consist of rows cross-referencing the keys of related objects.

The mathematical abstraction of the relational model has allowed the introduction of powerful and easy-to-use user languages for retrieval and updates of databases. It has also allowed the recent development of efficient implementations. The latter were facilitated by the invisibility to the user of the computer processing, which permits optimization without affecting the user. The semantic models offer a higher degree of abstraction. This results in much more concise user programs, as well as speedier processing due to optimization and other factors. Beyond that, the semantic models offer a plethora of other features.

The relational databases have provided a good service in many conventional database applications. However, in situations where the structure of information is complex, or where greater flexibility is required (objects with unknown identifiers, or objects moving from one category to another, etc.), or where non-conventional data is involved (long texts, images, etc.), other approaches need to be considered: semantic and/or object-oriented databases.

Akin to semantic models are object-oriented database models. They offer, to various degrees, many of the features of the semantic models, in the sense of abstracting information, and, in addition, formalize some behavioral properties of the data. We have shown that the latter behavioral properties can be easily added to semantic models when necessary, thus unifying the semantic and object-oriented approaches to databases. Our algorithms for efficient implementation of semantic databases models are applicable to object-oriented databases as well.

Storage of spatial data

As an example of a characteristic problem and an approach to its solution, let us consider observations of temperatures of the ocean. The ocean can be regarded as a four-dimensional Euclidean space of longitude, latitude, depth, and time. Thus, temperature T is a function $T(x,y,z,t)$. Additionally, there is a discrete dimension of observation sources, which may disagree between them. Thus, the temperature function may have five arguments: $T(x,y,z,t,s)$. A characteristic simple query, Q_1, to the system is to find the temperature for a given 5-dimensional point. A more complex query, Q_2, is to find the temperature of a four-dimensional space-time point independent of the observation source, obtained by weighing the different sources according to their known reliability, *etc.* Another complex query, Q_3, is to

find the average temperature of a given arbitrary space-time body. Another query, Q_4, interpolates data for unreported points.

We notice that the space-time contains infinitely many logical points. Therefore, both the observations and their database storage must represent a finite sampling of the space. Each measurement gives a temperature representing some (normally small) body of specified geometry (including location). For example, the geometry of one very simple body can be described as a hyper-rectangle of $1m \times 1m \times 1m \times 1hour$ whose edges are parallel to the axes and whose lowest point is (100,100,100,1994:12:31:12:00). Our first approximation of the relevant database schema fragment is:

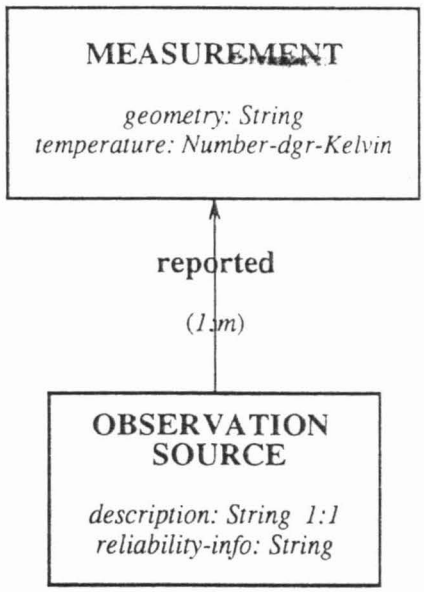

Figure 1. A simple schema fragment

Now, if we assume that there is a Boolean function *lies−in* that takes a point and a geometry and produces *true* if the point lies in the body, then the query Q_1 can be solved using any regular database language, for example the predicate calculus language of [Rishe-88-DDF]:

T(x,y,z,t,s)= get m.TEMPERATURE where
s REPORTED m and lies-in((x,y,z,t),m.GEOMETRY)

Although this query looks simple, if there are no restrictions on the kind of geometrical specifications, the run of this query will require searching through all the measurements of the given source and evaluating the function *lies−in* for each of them. This can take hours.

A more efficient solution can be obtained by representing the geometry as the smallest set of non-overlapping hyper-rectangles of various sizes. This solution will also eliminate the unknown function *lies−in*, as well as will allow averaging and other calculations.

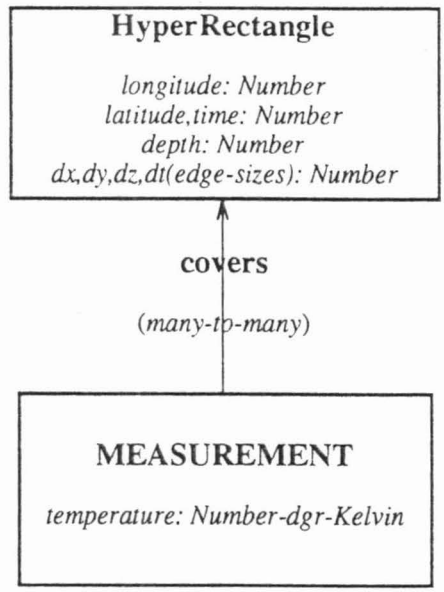

Figure 2. A replacement for the *geometry* attribute

For this schema, the query Q_1 becomes:
T(x,y,z,t,s)= get m.TEMPERATURE
where s REPORTED m and exists a HyperRectangle h such that m COVERS h and
h.LONGITUDE≤x≤h.LONGITUDE+h.DX and h.LATITUDE≤y≤h.LATITUDE+h.DY

and

h.DEPTH≤z≤h.DEPTH+h.DZ and
h.TIME≤t≤h.TIME+h.DT

If dx,dy,dz, and dt are known *a-priori*, then a database system utilizing the file management algorithm of [Rishe-89-EO] will normally solve this query, as well as the complex query Q_2, in just 5 disk accesses, *i.e.* in a tiny fraction of a second.

Another improvement is the representation of the space-time as a hexadecimal tree of hyper-quadrants. The applicability of this method to the storage of the *actual* unprocessed measurements is not fully clear yet, but it is a major improvement for the storage of derived/interpolated data which pertains to large continuous parts of the space-time. The space-time is bounded and, therefore, it can be embedded in a huge hyper-rectangle, S. Now, let us halve all the edges of S, thus partitioning S into 16 hyper-quadrants touching each other at the center of gravity of S. Let us denote them by the 16 hexadecimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f. Each of them can in turn be partitioned into 16 smaller hyper-quadrants, denoted by two hexadecimal digits, *e.g.* #7 is partitioned into 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 7a, 7b, 7c, 7d, 7e, 7f. Each of them can be further partitioned, and so on. Now, any continuous space-time can be represented, to any desired degree of precision, as a finite set of non-overlapping hyper-quadrants of various sizes (normally, various large hyper-quadrants covering the interior of the body, and small ones covering the boundary region). Thus, the geometry of a measurement can be represented by a multi-valued attribute *hyperquadrant*. Any point *(x,y,z,t)* can be represented by the smallest permitted hyper-quadrant, *microhyperquadrant*(x,y,z,t). Now, the query Q_1 becomes:

T(x,y,z,t,s)= get m.TEMPERATURE where s REPORTED m and (m has a hyper-quadrant which is a prefix of the hexadecimal string *microhyperquadrant*(x,y,z,t))

This query can be normally resolved in just 3 disk accesses without any *a-priori* knowledge. The queries Q_2, Q_3, and Q_4 also become very efficient. The storage requirements are small. Still further reduction in the storage can be obtained if we sample the space-time not into very small bodies of approximately constant temperature but into larger bodies whose temperature can be represented by an analytical function. For such bodies we will store the average temperature as well as a character string describing the offset in terms of the points' coordinates:

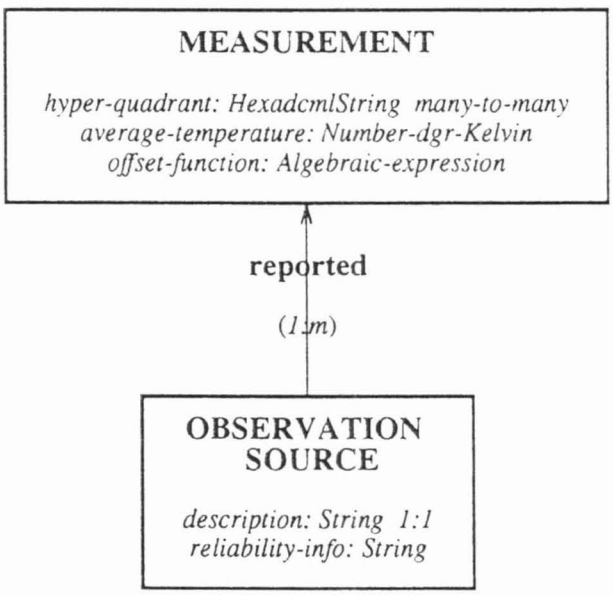

Figure 3. One measurement has several hyper-quadrants and a possible offset

Now, the query Q_1 becomes:
T(x,y,z,t,s)= get m.AVERAGE-TEMPERATURE+ (m.OFFSET-FUNCTION)(x,y,z,t) where s REPORTED m and m has a hyper-quadrant which is a prefix of the hexadecimal string *microhyperquadrant*(x,y,z,t).

Our idea of hyper-quadrants is derived from the quadrants used in GIS (geographic information systems), although there the quadrants are two-dimensional (longitude and latitude) and do not fully participate in a database schema. In the development of an ocean-geographic information system, one can use many existing results in GIS. A GIS is an automated system for the input, editing, management, display, and analysis of geographically referenced spatial data, along with associated non-spatial attribute data. For example, a GIS might contain map data of all real estate parcels in an area (spatial data) along with the address, the owner's name, and the tax rate (non-spatial attribute data) for each parcel. [Dangermond&Freedman-86] list a number of examples of GIS applications for municipal governments, such as making facility location decisions and performing vehicle routing, traffic analysis, land use planning, and facilities management. The U. S. Federal government agencies use GISs for a variety of purposes including maintaining census data and tracking the geographical distribution of disease statistics.

The geographic data which is stored in a GIS can be represented in the computer in two different ways: vector or raster (cell-based). The decision as to which representation is used is frequently a function of how the data were captured for use in the GIS. A vector representation stores the information in an explicit point, line, area format, with each object (e.g., a house, river, or field) "tagged" with its attributes. This format is used in GIS applications which require visually appealing output and/or precise locational information. Data in this format are usually captured by manual digitizing or scanning of existing map data. The raster format divides an area into a set of grid cells. These grid cells are generally of uniform size and shape, with the most common shape being a square. The attributes are associated with the individual grid cells. The individual objects no longer have a distinct identity, rather an object is implicitly defined in that all the cells of the object have the same attribute value. Satellite imagery is the most common type of data which is in a raster format. The relative advantages and disadvantages of these two formats for different applications have been discussed in [Maffini-87].

There are five basic deficiencies which exist in most contemporary GISs — as far as their appropriateness for the oceanographic data is concerned:

1. The GIS need only two dimensions: longitude and latitude; in oceanographic applications one uses four and five dimensional spaces.

2. Most GIS systems store the spatial data separately from the attribute data [Waugh&Healey-87], which can make the system inefficient and difficult to use.

3. Because of the distinct differences between vector and raster data, most operational GISs have been built around one or the other of these two types, but not both [Keating&*al.*-87]. If both types of data are required, two separate GISs must be used.

4. Many existing systems have been designed as stand-alone, special purpose systems with little attention paid to traditional database management concerns such as data protection, security and integrity [Frank-88].

5. GISs built using existing commercial database management systems as a central core, however, are not well suited to the requirements of GIS processing ([Frank-88],

[Keating&*al.*-87]). The primary reason for this is the poor performance of these systems in retrieving spatial data. The problem is that items that are located close together geographically are not stored physically close together on the computer disk. This results in inefficient access to these data.

Much of the research on GIS, which is useful to our project, in recent years has focused on quadtrees (trees of quadrants) ([Abel&Smith-86], [Anthony&Corr-88], [Bell&*al.*-88], [Gahegan-89], [Gargantini-82], [Ibbs&Stevens-88], [Jackson&Mason-86], [Jackson&*al.*-88], [Keating&*al.*-87], [Mark&Lauzon-84], [Orenstein&Manola-88], [Peuquet-84], [Peuquet-86], [Ripple&Ulshoefer-87], [Samet&*al.*-84], [Shaffer&Samet-87], [Shaffer&Samet-90], [Shaffer&*al.*-90], [Samet-84], [Samet&*al.*-86], [Waugh&Healey-87]) and on the use of the relational model ([Abel&Smith-86], [Abel-89], [Haralick-80], [Lorie&Meier-84], [Shapiro-80], [VanRoessel-87], [Waugh&Healey-87]). Deficiencies of the relational model for use in a GIS have been noted ([Keating&*al.*-87], [Lorie&Meier-84]). Research has also led to questioning traditional database models, including the relational model. Significantly, Webster [Webster-88] noted that "object-oriented models such as the semantic database models" will be important in "moving GIS technology forward from the limited database architecture to which it is tied."

In our project we expect to use a multi-dimensional generalization of linear quadtrees in two ways. First, a linear quadtree address can used as the primary spatial index into the database. Use of the address in this manner will ensure that measurements close in space-time will be stored physically close on the computer disk. Second, linear quadtrees will also be used as the storage structure for the raster data. When a raster image is first brought into the system, the image would be processed to break it into its discrete area components. These discrete areas would then be stored as sets of linear quadtree addresses. A critical portion of this design is the use of a linear quadtree address as the spatial search key. A standard quadtree utilizes pointers to traverse the tree, but [Gargantini-82] described a pointerless quadtree which she termed a "linear quadtree." It is this linear quadtree key which is most useful in the implementation of our schema. Her algorithm effectively maps the two dimensional space into one dimension, and we generalize it to multi-dimensional input. The use of quadtrees as integrated spatial indexes, rather than explicit storage structures, has been advocated in [Ibbs&Stevens-88] and [Jackson&*al.*-88]. Using the semantic database model of [Rishe-88-DDF] and its implementation algorithms of [Rishe-89-EO] in conjunction with the linear quadtree addresses, we will be able to provide efficient access to the spatial data.

References

[Abel-89] D.J. Abel. SIRO-DBMS: A Database Tool-Kit for Geographical Information Systems. Int. J. Geographical Information Systems, vol. 3, no. 2, 1989, pp. 103-116.

[Abel&Smith-86] D.J Abel and J.L. Smith. ''A Relational GIS Database Accomodating Independent Partitionings of the Region.'' *Proceedings: Second International Symposium On Spatial Data Handling,* 1986, pp. 213-224.

[Anthony&Corr-88] S.J Anthony and D.G Corr. ''Data Structures in an Integrated Geographical Information System.'' *ESA Journal,* vol. 12, no.1, 1988, pp. 69-72.

[Bell&*al.*-88] S.B.M. Bell, B.M. Diaz, and F.C. Holroyd. Capturing Image Syntax Using Tesseral Addressing and Arithmetic. In J. Muller (ed.), Digital Image Processing in

Remote Sensing, Taylor & Francis, London, 1988.

[Dangermond&Freedman-86] J. Dangermond and C. Freedman. Findings Regarding a Conceptual Model of a Municipal Database and Implications for Software Design. Geo-Processing, vol. 3, 1986, pp. 31-49.

[Frank-88] A.U. Frank. Requirements for a Database Management System for a GIS. Photogrammetric Engineering and Remote Sensing, vol. 54, no. 11, 1988, pp. 1557-1564.

[Gahegan-89] M.N. Gahegan. An Efficient Use of Quadtrees in a Geographical Information System. Int. J. Geographical Information Systems, vol. 3, no. 3, 1989, pp. 201-214.

[Gargantini-82] I. Gargantini. An Effective Way to Represent Quadtrees. Communications of the ACM, vol. 25, no. 12, 1982, pp. 905-910.

[Haralick-80] R.M. Haralick. A Spatial Data Structure for Geographic Information Systems. In H. Freeman and G.G. Pieroni (eds.), Map Data Processing, Academic Press, New York, 1980.

[Ibbs&Stevens-88] T.J. Ibbs and A. Stevens. Quadtree Storage of Vector Data. Int. J. Geographical Information Systems, vol. 2, no. 1, 1988, pp. 43-56.

[Jackson&*al.*-88] M.J. Jackson, W.J. James, and A. Stevens. The Design of Environmental Geographic Information Systems. Philosophical Transactions of the Royal Society of London, series A, vol. 324, 1988, pp. 373-380.

[Jackson&Mason-86] M.J. Jackson and D.C. Mason. The Development of Integrated Geo- Information Systems. Int. J. Remote Sensing, vol. 7, no. 6, 1986, pp. 723- 740.

[Keating&*al.*-87] T. Keating, W. Phillips, and K. Ingram. "An Integrated Topologic Database Design for Geographic Information Systems." *Photogrammetric Engineering and Remote Sensing,* vol. 53, no. 10, 1987, pp. 1399-1402.

[Lorie&Meier-84] R.A. Lorie and A. Meier. Using a Relational DBMS for Geographical Databases. Geo-Processing, vol. 2, 1984, pp. 243-257.

[Maffini-87] G. Maffini. Raster Versus Vector Data Encoding and Handling: A Commentary. Photogrammetric Engineering and Remote Sensing, vol. 53, no. 10, 1987, pp. 1397-1398.

[Mark&Lauzon-84] D.M. Mark and J.P. Lauzon. Linear Quadtrees for Geographic Information Systems. Proceedings of the International Symposium On Spatial Data Handling, Volume II, 1984, pp. 412-430.

[Orenstein&Manola-88] J.A. Orenstein and F.A. Manola. PROBE Spatial data Modeling and Query Processing in an Image Database Application. IEEE Transactions on Software Engineering, vol. 14, no. 5, 1988, pp. 611-629.

[Peuquet-84] D.J. Peuquet. Data Structures for a Knowledge-Based Geographic Information System. Proceedings of the International Symposium On Spatial Data Handling, Volume II, 1984, pp. 372-391.

[Peuquet-86] D.J. Peuquet. The Use of Spatial Relationships to Aid Spatial Database Retrieval. Proceedings: Second International Symposium On Spatial Data Handling, 1986, pp. 459-471.

[Ripple&Ulshoefer-87] W.J. Ripple and V.S. Ulshoefer. Expert Systems and Spatial Data Models for Efficient Geographic Data Handling. Photogrammetric Engineering and Remote Sensing, vol. 53, no. 10, 1987.

[Rishe-88-DDF] N. Rishe. *Database Design Fundamentals: A Structured Introduction to Databases and a Structured Database Design Methodology.* Prentice-Hall, Englewood Cliffs, NJ, 1988. 436 pp.

[Rishe-89-EO] N. Rishe. "Efficient Organization of Semantic Databases" *Foundations of Data Organization and Algorithms.* (FODO-89) W. Litwin and H.-J. Schek, eds. *Springer-Verlag Lecture Notes in Computer Science*, vol. 367, pp. 114-127, 1989.

[Rishe-91-FS] N. Rishe. "A File Structure for Semantic Databases." *Information Systems*, 16, *4* (1991), pp. 375-385.

[Rishe-92-DDS] N. Rishe. *Database Design: The Semantic Modeling Approach.* McGraw-Hill, 1992, 528 pp.

[Rishe-93-MT] N. Rishe. "A Methodology and Tool for Top-down Relational Database Design." *Data and Knowledge Engineering*, 10 (1993) 259-291.

[Samet-84] H. Samet, "The quadtree and related hierarchical data structures," *Computing Surveys* 16, 1984, 187-260.

[Samet&*al.*-84] H. Samet, A. Rosenfeld, C.A. Shaffer, and R.E. Webber. Use of Hierarchical Data Structures in Geographical Information Systems. Proceedings of the International Symposium On Spatial Data Handling, Volume II, 1984, pp. 392-411.

[Samet&*al.*-86] H. Samet, C.A. Shaffer, R.C. Nelson, Y. Huang, K. Fujimura, and A. Rosenfeld. Recent Developments in Quadtree-Based Geographic Information Systems. Proceedings: Second International Symposium On Spatial Data Handling, 1986, pp. 15-32.

[Shaffer&*al.*-90] C.A. Shaffer, H. Samet, and R.C. Nelson. QUILT: A Geographic Information System Based on Quadtrees. Int. J. Geographical Information Systems, vol. 4, no. 2, 1990, pp. 103-131.

[Shaffer&Samet-87] C.A. Shaffer and H. Samet. Optimal Quadtree Construction Algorithms. Computer Vision, Graphics, and Image Processing, vol. 37, 1987, pp. 402-419.

[Shaffer&Samet-90] C.A. Shaffer and H. Samet. Set Operations for Unaligned Linear Quadtrees. Computer Vision, Graphics, and Image Processing, vol. 50, 1990, pp. 29-49.

[Shapiro-80] L.G. Shapiro. Design of a Spatial Information System. In H. Freeman and G.G. Pieroni (eds.), Map Data Processing, Academic Press, New York, 1980.

[VanRoessel-87] J.W. Van Roessel. Design of a Spatial Data Structure Using the Relational Normal Forms. Int. J. Geographical Information Systems, vol. 1, no. 1, 1987, pp. 33-50.

[Waugh&Healey-87] T.C. Waugh and R.G. Healey. The GEOVIEW Design: A Relational Data Base Approach to Geographical Data Handling. Int. J. Geographical Information Systems, vol. 1, no. 2, 1987, pp. 101-118.

[Webster-88] C. Webster. Disaggregated GIS Architecture: Lessons From Recent Developments in Multi-Site Database Management Systems. Int. J. Geographical Information Systems, vol. 2, no. 1, 1988, pp. 67-79.