AN EXPERIMENTAL RELATIONAL DATABASE SYSTEM FOR CARTOGRAPHIC APPLICATIONS

Prashant D. Vaidya
Linda G. Shapiro
Robert M. Haralick
Gary J. Minden*

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

ABSTRACT
Cartographic information includes not only points, lines, and areas, but also relationships among them. We have proposed for this purpose an entity-oriented relational database system (Shapiro and Haralick, 1980). Each geographic entity such as a region or river or city is represented by a set of relations describing its properties, its parts, its related entities, and all the relationships among them. In this paper we describe our first experimental cartographic database software system employing these concepts to store and retrieve watershed data for a portion of the state of Virginia.

INTRODUCTION

Most geographic information systems have been organized either as a set of polygons or as a raster of grid cells. The polygon representation is suitable for explicitly representing region boundaries and line data such as rivers or roads. One large system using a polygon representation in a topologically consistent way is the U. S. Census DIME System (Cook and Maxfield, 1967). In the raster representation, a regular grid is placed over a map, and certain properties such as population, major crop, or land type are recorded for each grid cell. The raster representation retains the spatial relationships of the map, but efficiency is sacrificed for some operations (see Pequet, 1979).

In a previous paper (Shapiro and Haralick, 1980), we suggested a relational approach to designing a spatial information system. Our approach has the advantage of allowing either vector or raster data or both in a unified framework suitable for high-level query. This paper describes the use of this approach in a first experimental database system for storage and retrieval of watershed data. We first review the definition of the general spatial data that is the building block of the system. Next we describe the logical structure of the experimental database. Finally we describe the stack-oriented query language that is used to

*Department of Electrical Engineering, University of Kansas

133
communicate with the database system.

**THE SPATIAL DATA STRUCTURE**

In this section we define a general spatial data structure that can be used to represent any spatial information or relational data.

An atom is a unit of data that will not be further broken down. Integers and character strings are common examples of atoms. An attribute-value table $A/V$ is a set of pairs $A/V = \{(a,v) \mid a$ is an attribute and $v$ is the value associated with attribute $a\}$. Both $a$ and $v$ may be atoms or more complex structures. For example, in an attribute-value table associated with a structure representing a person, the attribute $AGE$ would have a numeric value and the attribute $MOTHER$ might have as its value a structure representing another person.

A spatial data structure $D$ is a set $D = \{R_1, \ldots, R_K\}$ of relations. Each relation $R_k$ has a dimension $N_k$ and a sequence of domain sets $S(1,k), \ldots, S(N_k,k)$. That is for each $k = 1, \ldots, K$, $R_k \subseteq S(1,k) \times \ldots \times S(N_k,k)$. The elements of the domain sets may be atoms or spatial data structures. Since the spatial data structure is defined in terms of relations whose elements may themselves be spatial data structures, we call it a recursive structure. This indicates 1) that the spatial data structure is defined with a recursive definition, 2) that it will often be possible to describe operations on the structure by simple recursive algorithms, and 3) that it can naturally represent both relational and hierarchical dependencies.

A spatial data structure represents a geographic entity. The entity might be as simple as a point or as complex as a whole map. An entity has global properties, component parts and related geographic entities. Each spatial data structure has one distinguished binary relation containing the global properties of the entity that the structure represents. The distinguished relation is an attribute-value table and will generally be referred to as the $A/V$ relation. When a geographic entity is made up of parts, we may need to know how the parts are organized. Or, we may wish to store a list of other geographic entities that are in a particular relation to the one we are describing. Such a list is just a unary relation, and the interrelationships among the parts are $n$-ary relations.

**THE LOGICAL DATABASE STRUCTURE**

One objective of a database system is to systematize the access to the data elements. The first step in the implementation of any database management system is the design of its conceptual model (also known as a data model). The conceptual model is a representation of the entire information content of the database, in a form that is somewhat abstract in comparison with the way in which the data is physically stored (Date, 1977). In order to translate a model into an
operational system, the model has to be described in a form which lends itself to implementation. Such a description is called a schema (Wiederhold, 1977). A schema defines the logical structure of the database without any storage/access details.

In this section we develop the schema for our geographic database system. The spatial data structure is the primitive or the building block of the system. Each spatial data structure represents one particular type of spatial data and has a unique name. The schema is developed in the form of a prototype structure for each type of spatial data structure. The prototype indicates what attributes may be found in the A/V relation of this type of spatial data structure and what relations besides the A/V relation comprise the data structure.

The data* used in this system is of two types: stream data and road data. The stream data consists of watershed areas, water streams, and labels, while the road data consists of a road network.

A digitized map** of the stream data, along with a description of symbols used in the map to represent various entities, is shown in Figure 1. The stream data comes from the region labeled N3 in Figure 1. Region N3 is a watershed area.

The road data used is a subset of the road data for the entire Appalachia quadrangle which includes region N3. The road network is similar to the stream network. There are two types of roads: primary and secondary. The roads may intersect with roads of the same type or of a different type, but unlike streams, the roads may cross the boundaries of regions.

From the description of the data, it can be observed that the basic geographic entities used in the system are regions, water streams, roads, and labels. A region can be represented by a polygon which has a closed boundary. A stream or a road can be represented by a chain which is comprised of an ordered list of points. A label can be represented by a point which has coordinates. Thus we have the following high level spatial data structure types: 1) REGION, 2) WATER STREAMS, 3) STREAM, 4) ROAD NETWORK, 5) ROAD, and 6) LABEL. The low level spatial data structure types are: 1) POLYGON and 2) CHAIN. A POINT is implemented as an atom.

Figure 2 illustrates the prototypes REGION, WATER STREAMS,

* This data was obtained from the Dept. of Fisheries and Wildlife Science, Virginia Tech, Blacksburg, VA, courtesy of Dr. Robert Giles.
** This map is a subset of the Watershed Area Map for the Appalachia Quadrangle, located in WISE county, VA. For more information refer to the U.S.G.S. map number N3652.5 - W8245/7.5.
Figure 1 shows a digitized map of the stream data used in our system.
Figure 2 illustrates the prototypes for spatial data structures REGION, WATER STREAMS, STREAM, LABEL, POLYGON, and CHAIN.
STREAM, LABEL, POLYGON, and CHAIN. Each spatial data structure of type REGION consists of four relations: i) the A/V relation, A/V REGION, ii) SUBREGION ADJACENCY, iii) STREAM NETWORK, and iv) LABELS. The A/V relation has four attributes: NAME whose value is a character string representing the name of the region, AREA whose value is a number representing the area of the region, BOUNDARY whose value is a spatial data structure of type POLYGON (to be described later) representing the boundary of the region, and PARENT whose value is a spatial data structure which itself is of type REGION, representing the next immediate region which encloses the region under consideration.

A region may have to be divided into subregions, in which case the subregions are stored in a SUBREGION ADJACENCY relation. This is a binary relation associating each subregion with every other subregion that neighbors it. Both the components of each pair in the relation are spatial data structures of type REGION.

The relations of type STREAM NETWORK are unary relations whose components are spatial data structures of type WATER STREAMS. The relations of type LABELS are unary relations whose components are spatial data structures of type LABEL. There are two types of streams: ephemerals and perennialials. WATER STREAMS therefore consists of two relations: EPHEMERALS and PERRINALS. Both EPHEMERALS and PERRINALS are unary relations whose components are spatial data structures of type STREAM.

Each spatial data structure of type STREAM consists of two relations: an A/V relation called A/V STREAM and a binary relation INTERSECTING STREAMS. The A/V STREAM relation has seven attributes: NAME, TYPE, and ORDER whose values are simple character strings representing the name of the stream, its type, and its order, respectively; LENGTH, # INTERSECTING EPHEMERALS, and # INTERSECTING PERRINALS whose values are numbers representing the length of the stream, the number of ephemerals intersecting, and the number of perennialials intersecting, respectively; and COURSE whose value is a spatial data structure of type CHAIN representing the course of the stream.

The relations of type INTERSECTING STREAMS are binary relations whose components represent the point of intersection, which is an atomic POINT and the stream intersecting at that point, which is a spatial data structure of type STREAM.

Each spatial data structure of type LABEL consists of only one relation, an A/V relation called A/V LABEL. The A/V relation in this case has two attributes: NAME whose value is a character string representing the name of the label and LOCATION whose value is an atomic POINT representing the location of the label. The labels in our experimental system would be replaced by other point data in a real system.

The low level spatial data structure types include the POLYGON and CHAIN and also the atom POINT. We represent the boundary of any region by a spatial data structure of type POLYGON. A polygon is comprised of chains. Each spatial
data structure of type POLYGON has a unary relation called CHAINS, whose components are spatial data structures of type CHAIN.

We represent the course of any water stream or road by a spatial data structure which is of type CHAIN. Each spatial data structure of type CHAIN is comprised of two relations: an A/V relation called A/V CHAIN and a relation POINTS. A chain has a region to its left and region to its right. The A/V relation therefore has two attributes: LEFT and RIGHT. The values of both these attributes are spatial data structures of type REGION.

The relation POINTS is an ordered list (a binary relation) of points that define the chain. A POINT is an atom, a data element at the innermost level which can not be further broken down. A POINT consists of an ordered pair (X, Y) where X represents the latitude or the X co-ordinate and Y the longitude or the Y co-ordinate.

ACCESSING THE DATABASE

Most relational database systems include a relational query language through which a user can interact with the database. For example, in the SEQUEL language (Chamberlin and Boyce, 1974), a user might specify the command

```
SELECT SUPPLIER NUMBER, STATUS
FROM SUPPLIER RELATION
WHERE CITY = 'BLACKSBURG'
AND STATUS > 20.
```

This command instructs the system to go to the relation whose name is SUPPLIER RELATION, find those tuples having 'BLACKSBURG' in the CITY component and a number > 20 in the STATUS component, and construct a new relation consisting of the SUPPLIER NUMBER and STATUS components of the selected tuples.

Because our system is entity-oriented rather than relation-oriented, languages like SEQUEL are only indirectly applicable. We envision, in the future, an intelligent system with natural language query facilities so that a user might make the request:

```
FIND ALL RIVERS
WITHIN 200 MILES OF ROANOKE,
LONGER THAN 50 MILES, AND
CROSSED BY INTERSTATE 81.
```

The intelligent system, using knowledge of the prototypes of the spatial data structures and the semantics of the various tuple components, would invoke a deduction system that would determine the best sequence of relational operations required to extract this information from the database. A lot of our work involves the design of this intelligent component.

The current experimental database system uses a
stack-oriented query language similar to the FORTH language (Moore, 1974). Users can define constants, variables, and arrays and perform arithmetic operations or database operations using control structures. New commands can be defined via a simple macro-definition facility. The system itself is viewed as a calculator with a large stack. The arguments of all operations are performed on the top entry or top n entries of the stack, where n depends on the particular operation. Results are returned to the stack. To give the reader a feel for the power of the query language, we will briefly describe the primitive operations and give an example of their use.

The query language interpreter supports the high-level data types integer, boolean, character-string, point, spatial data structure, and relation. Standard arithmetic and relational operations are provided. The type of the top stack element (TOS) can be determined by test commands. The stack elements can be manipulated, regardless of content, by such operations as SWAP (exchange TOS with next element in stack) DROP (remove TOS), DUP (duplicate TOS), ROT (rotate top three stack elements), -ROT (reverse rotate top three stack elements), and other similar operations.

The stack-oriented query language program control primitives include the IF-ELSE-THEN construct for conditional execution and the DO-+LOOP construct and REPEAT-UNTIL construct for iteration. The vocabulary manipulation primitives include the DEFINE-END DEF construct for defining new commands, the CONSTANT and VARIABLE commands for defining new constants and variables, the FETCH and STORE commands for adding to and removing from the stack selected vocabulary entries, and the ALLOCATE and FORGET commands for requesting and freeing vocabulary space.

The database manipulation primitives ALLOC_RDS and ALLOC_REL allow the user to allocate and catalog new spatial data structures and relations. The command FIND allows the user to locate structures by alphanumeric name. The commands LIST_RDS, LIST_REL, XLIST_RDS, and XLIST_REL allow the headers and contents of spatial data structures and relations to be listed.

The primitives NT_ATTACH and REL_ATTACH allow the user to add N-tuples of data from the stack to a relation and add a relation to a spatial data structure, respectively. The command INRELA? allows the user to check whether a specified N-tuple is in a given relation. The commands [NAME], [TYPE], [DIMEN], [LENGTH], and [USE_CNT] allow the user to request that certain attributes of a spatial data structure or relation be returned to the stack. The command [STRUCT] returns to the stack a pointer to the beginning of a structure, and the command [LINK] allows the user to advance the pointer through the structure, while the command [DATA] returns the data from the record pointed to.

The user can also request the loading and unloading (DB_LOAD and DB_UNLOAD) of the database, input and output from files (INPUT and OUTPUT), or print a portion of the vocabulary (DUMP). The session is terminated by the command DONE.
short example will illustrate the use of the query lan-
geage. The example session creates a new binary relation 
 V.Region X, inserts N-tuples into it, and attaches that 
relation to an existing spatial data structure Region X 
sing the query language commands which are marked by asteri-
sks here.

<table>
<thead>
<tr>
<th>Input by the User</th>
<th>Explanation and Action Taken</th>
</tr>
</thead>
</table>
| 2                 | Dimension of the relation is 
going to be the integer 2. |
| 2                 | Type of the relation is 
going to be the integer 2. |
| " A/V_REGION_X"   | Name of the relation is 
going to be A/V_REGION_X. |

<table>
<thead>
<tr>
<th>CHAR</th>
<th>A/V_REGION_X</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT</td>
<td>2</td>
</tr>
<tr>
<td>INT</td>
<td>2</td>
</tr>
</tbody>
</table>

Contents of Stack

*ALLOC_REL Allocates a relation header with the name A/V_REGION_X, 
dimension two, and type two, and puts a pointer to it on 
the top of the stack.

<table>
<thead>
<tr>
<th>REL</th>
<th>Contents of Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Header of A/V_REGION_X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Contents of Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; AREA&quot;</td>
<td>An attribute to be part of the A/V relation.</td>
</tr>
<tr>
<td>12345</td>
<td>The value of the attribute.</td>
</tr>
<tr>
<td>*ROT</td>
<td>Bring the pointer to the relation to the top of the stack.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REL</th>
<th>Contents of Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Header of A/V_REGION_X</td>
</tr>
<tr>
<td></td>
<td>CHAR</td>
</tr>
<tr>
<td>INT</td>
<td>12345</td>
</tr>
</tbody>
</table>

*NT_ATTACH Attach the N-tuple consisting of attribute and value to the relation. The stack becomes empty.

<table>
<thead>
<tr>
<th></th>
<th>Contents of Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot; A/V_REGION_X&quot;</td>
<td>Restore pointer to the relation A/V_REGION_X to the top of the stack.</td>
</tr>
<tr>
<td>*FIND</td>
<td>Name of the spatial data structure to which the relation is to be attached.</td>
</tr>
<tr>
<td>*DROP</td>
<td>Search the dictionary for the given spatial data</td>
</tr>
</tbody>
</table>
structure name and put a
pointer to it on the stack.

*DROP
Drop the top of the stack
which indicated the status
of the search.

SDS
REL
\rightarrow \text{Header of REGION X}
\rightarrow \text{Header of A/V REGION X}

Contents of Stack

*REL_ATTACH
Attach the relation to the
spatial data structure.

CONCLUSIONS

The experimental system has so far only demonstrated that
géographic data can be represented in the entity-oriented
relational framework. The next step in our work is to
implement high-level geographic algorithms using the primi-
tive structural operations now provided and to evaluate the
performance of these algorithms. The stack-oriented query
language is only a first step towards communication between
human and database system. Thus we can only conclude that
there is much more to do.

REFERENCES

English Query Language", Proc 1974 ACM SIGMOD Workshop
on Data Description, Access, and Control.

Cook, D. and W. Maxfield, "The Development of a Geographic
Base File and Its Uses for Mapping", Proceedings of

Date, An Introduction to Database Systems, 2nd Edition,

Moore, C. H., "FORTH, A New Way to Program a Computer",
Astronomy and Astrophysics Supplement, 1974, No. 15,
pp. 497-511.

Pequet, D. J., "A Raster Mode Algorithm for Interative
Modification of Line Drawing Data", Computer Graphics
and Image Processing 10, pp. 142-158.

Shapiro, L. G. and R. M. Haralick, "A Spatial Data Struc-

Wiederholdt, G., Database Design, McGraw-Hill, New York,
1977.