Managing Method of Spatial Data in Relational Database Management Systems*

ZHU Tie-wen[1,2], ZHONG Zhi-nong[1], JING Ning[1]

[1](*School of Electrical Science and Engineering, National University of Defence Technology, Changsha* 410073, *China*);

[2](*The First Aeronautical Institute of Air Force, Zhengzhou* 464000, *China*)

E-mail: twzhu@21cn.com

http://www.nudt.edu.cn

Received March 6, 2001; accepted July 12, 2001

Abstract: Today, data are stored in and managed by a DBMS(database management system). Relational database is the current database technique to solve the problem in the flat-file and hierarchical database model. In the application of GIS(geographical information system), it recurs to a mechanism i.e. so-called Spatial Database for arranging, analyzing and viewing spatial data. Because spatial database contains many different data formats and structures, these complex data lead to that spatial data manipulation and process may be very complex and difficult and tend to make mistakes. In this paper, we advise a new method by using relational database to manipulate spatial data. The scheme is to introduce the concept of RSDD (i.e., Regularly Spatial Discrete Domains), then define concepts about RPO (i.e., RSDD_based Primary Object) and RO (i.e., RSDD_based Object). The concept of RSDD can solve the conflict between the infinite precision real numbers of spatial object and the finite precision number systems of computers.

Key words: relational database; spatial database; RSDD; GIS

Geographical information systems (GISs) support applications that manipulate geographical data (or spatial data), such as urban planning, traffic control or natural resources management. Among the main issues in GIS design are the management of large amounts of data and the coexistence of two kinds of data: alphanumerical data and spatial data (geometry and topology). In addition, spatial data typically has extremely complex and variable structure, and it must be manipulated by specific operations.

GIS is primarily associated with spatial data, and therefore a large amount of the research effort in databases for GIS is related to spatial structures and access methods[1~4]. But, there is no consensuses on which are the appropriate methods to implement them into a DBMS. Today, data are stored in and managed via a Database Management System (DBMS)[5,6]. Relational database[7] is the current database technique to solve the problem in the flat-file and hierarchical database model. Relational DBMS (RDBMS) probably has the simplest structure a database can have. Data is organized in tables and tables contain records that consist of fields. GIS recurs to a mechanism, i.e., so-called Spatial Database, for aiming at storing, retrieving, manipulating, querying, and analyzing

* Supported by the Young Faculty Research Foundation of Ministry of Education of China (国家教育部优秀年轻教师基金)

ZHU Tie-wen was born in 1962. He is a vice-professor and Ph.D. candidate at the School of Electrical Science and Engineering, National University of Defence Technology. His research interests are geographical information system and database technique. **ZHONG Zhi-nong** was born in 1975. He is a Ph.D. candidate at the School of Electrical Science and Engineering, National University of Defence Technology. His research interests are GIS and database. **JING Ning** was born in 1963. He is a professor and doctoral supervisor of the School of Electrical Science and Engineering, National University of Defence Technology. His current research areas are GIS, database systems, GIS/Database applications for Internet information services.

spatial data. Spatial data are any data whose the underlying frame of reference is the Earth's surface, such as point, line, surface, and solid.

In the spatial database, a fundamental idea is how to represent geometry. But spatial database contains many different data formats and structures. These complex data lead to the fragmental data structures and loose relationships among themselves. Spatial data manipulation and process may be very complex and difficult and tend to make mistakes.

We attempt to manipulate the RSDD-based data types which we defined using relational database. This method can employ the mature technology of relational database to solve the complex problem of spatial database. This paper is organized as follows. In section 1, we briefly explain major concepts about GIS data structures. We introduce the RSDD-based spatial data type in section 2. Section 3 expresses the idea of using relational database to manipulate spatial data. Finally, we present our conclusions in section 4.

1 Spatial Data Type and Structure

The design of spatial data types and structures should be based on an understanding of the properties of spatial data in order to accommodate the design process as close as possible to the nature of spatial phenomena in the physical world. Most data manipulated by GIS are spatial data. Spatial data[8] consist of spatial objects made up of points, lines, surfaces, solids, and even data of higher dimension which include time. Examples of spatial data include cities, rivers, roads, counties, states, crop coverage, mountain ranges, etc. Often it is also desirable to attach non-spatial attribute information[9]. Examples of spatial properties include the extent of a given river, or the boundary of a given county, elevation heights, city names, etc. Spatial databases facilitate the storage and efficient processing of spatial and non-spatial information ideally without favoring one over the other. Such databases are found increasingly usage in environmental monitoring, space, urban planning, resource management, and etc.

Spatial data has two models[8]: raster model and vector model.

The raster data model divides our terrain into cells of equal and regular size and shape. Then, the value of any cell for a particular attribute is the value of the attribute in that cell. This division of the terrain into regular cells is called "regular tessellation". The cells can be of any shape as long as all of them are of the same shape. Thus, we could have a grid of triangular, hexagonal, or square cells. For our particular example, suppose we are assigned to store data concerning the elevation and temperature at different part of a given terrain or geographical region, which shape we choose would depend on how the attributes, i.e. elevation and temperature, vary over the terrain. If for some reason we know that temperature seems to be more or less constant over a hexagonal region, we could choose a hexagonal shape as our cell shape.

In our example, we have more than one attribute about which we want information. One good way to visualize this model would be to think of a (considerably transparent) quilt laid over our terrain. The quilt is composed of equal and regular cells. This particular quilt may represent information of elevation, so each cell in this quilt is labeled with the value of elevation in that cell. For temperature, we have yet another quilt, or "layer". This quilt is divided into cells exactly like the first quilt. The only difference between these two quilts is the attribute whose value is stored in each cell. In other words, we have different layers representing different attributes, and each layer is stored in a different file. Each cell in each quilt can only be assigned one value at a time. This rule makes sense if we realize how one cell cannot have two different elevations.

Each cell is called a "pixel" (picture element). The area of each cell or pixel is called its "spatial resolution". The spatial resolution tells us about the accuracy for this particular choice for cell size. In other words, if we have smaller cells, we would probably have more accuracy because we would be able to better capture or approximate the difference between attribute values over our terrain. A minimum mapping unit is "the smallest element we can

uniquely represent in our data" [3,4,10].

The vector model, like the raster model, divides space into discrete elements, but unlike the raster model, it performs this division based on geographic features and position rather than on the basis of a uniform grid. Since the partitions this model imposes on our terrain are not necessarily of equal or regular size, the division of the terrain space is referred to as an "irregular tessellation".

There seems to be a hierarchy of four basic partitions in the vector model: point, line, plane and solid. All four of these objects are defined using the standard Cartesian coordinate system. A point is thus represented by a single coordinate tuple (x,y,z). A line is a sequence of these points which forms a straight edge. A line has length and direction and is represented as a sequence of coordinate tuples. A surface (surface in plane is area or polygon) could probably best be described as a "simple cycle" bounding a region. A simple cycle is a sequence of points where the start and the end of the sequence are the same point. A solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces.

2 RSDD-Based Spatial Data Type

Most spatial databases cannot manage vector data and raster data in a uniform way. Markus Schneider has introduced *realm* concept in Ref.[11]. A realm is a set of points and non-intersecting line segments over a discrete domain. Realm-based spatial data types are called POINTS, LINES, and REGIONS. Realm can successfully manipulate 2-dimensional spatial data. It can enforce geometric consistency of related spatial objects, guarantee nice closure properties for the computation with spatial data types, shield geometric computation in query processing from numeric correctness and robustness problems, and be used as an index into the database[12]. But, realm does not deal with 3-dimensional spatial data. In order to manage 3D data, we extend realm concept to 3-dimensional space, called RSDD(Regularly Spatial Discrete Domains). The topological relations and the operations defined on 3D space are more complex, thus the extending of RSDD from 2D to 3D should not be a simple extension.

With regard to terminology and definition of topological relation of 3D spatial object, please refer to Ref.[13].

Now, we define a Regularly Spatial Discrete Domain (RSDD) and RSDD_based Primary Objects (RPOs), such as RPO-point, RPO-line, RPO-plane and RPO-solid, as well as some predicates and operations on them. All definitions are based on error-free integer arithmetic that enables direct and robust implementation. Let $N = \{0, ..., n\text{-}1\}$, 0, ..., n-1 be integers. As depicted in Fig.1, the *RSDD* is a set of points in $N\times N\times N$. A point in RSDD is called *RPO-point*, i.e., a *RPO-point* is a tuple $(x, y, z)\in N\times N\times N$. A *RPO-line* is a segment whose ends belong to $N\times N\times N$. A *RPO-plane* is a polygon composed of coplanar RPO-lines and includes inside this polygon. A *RPO-solid* is a solid whose boundary consists of RPO-planes.

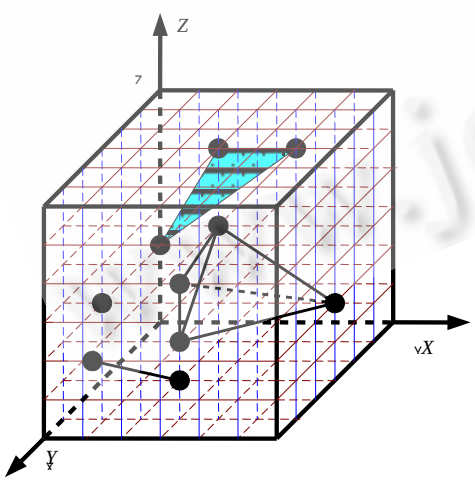


Fig.1 RSDD and RSDD-based Primary Objects

Given N, let PO_N be a set of all RPO-points, LI_N a set of all RPO-lines, PL_N and SO_N are set RPO-planes and RPO-solids, respectively. A RSDD-based Object is a set $RO = PO_{sub} \cup LI_{sub} \cup PL_{sub} \cup SO_{sub}$ such that:

(1) $PO_{sub} \subseteq PO_N, LI_{sub} \subseteq LI_N, PL_{sub} \subseteq PL_N, SO_{sub} \subseteq SO_N$;

(2) $\forall l \in LI_{sub}: l = (p_1, p_2), p_1 \in PO_{sub} \wedge p_2 \in PO_{sub}$;

(3) $\forall p \in PO_{sub}, l \in LI_{sub:} \neg(p \text{ in } l)$;

(4) $\forall l \in LI_{sub}, s \in PL_{sub:}$ when $\neg(l \text{ in } s) \wedge$have intersection,(intersection of l and s) $\in PO_{sub;}$

(5) $\forall l \in LI_{sub}, v \in SO_{sub:}$ when $\neg(l \text{ in } v) \wedge$have intersection,(intersection of l and v) $\in PO_{sub;}$

(6) $\forall s \in PL_{sub}, v \in SO_{sub:}$ when $\neg(s \text{ in } v) \wedge$have intersection,(intersection of s and v) $\in PO_{sub} \cup LI_{sub}$;

(7) $\forall l_1, l_2 \in LI_{sub:} \neg(l_1 = l_2) \wedge \neg(l_1$ and l_2 intersect) $\wedge \neg(l_1$ and l_2 overlap);

(8) $\forall s_1, s_2 \in PL_{sub:} \neg(s_1 = s_2) \wedge \neg(s_1$ and s_2 intersect) $\wedge \neg(s_1$ and s_2 overlap);

(9) $\forall v_1, v_2 \in SO_{sub:} \neg(v_1 = v_2) \wedge \neg(v_1$ and v_2 meet in plane) $\wedge \neg(v_1$ and v_2 overlap);

Intuitively, a *RO* is a set that composed of RPOs. Its points locate in grid points of RSDD. The two ends of its lines also locate in RSDD and there are no other RPO-points on the lines except ends. For the relation of RPO-lines and PRO-planes, either there is no intersection or their intersection is a RPO-point. For the relation of RPO-lines and PRO-solids, either there is no intersection or their intersection is a RPO-point. For the relation of RPO-planes and PRO-solids, there is no intersection or their intersection is a RPO-point, or RPO-lines. For the relation of two RPO-lines or two RPO-planes, they are not equal to each other and there is no intersection and overlap. For the relation of two RPO-solids, they are not equal to each other and do not meet in plane and overlap.

With the above assumption, we can manipulate the spatial object only to operate on ROs. For manipulating ROs, we need three groups of operations. The first group of operations contain Insert, Split and Delete, which are fundamental operations on ROs. The Insert operation takes a RO and a RPO as operands. The Delete operation takes a RO and the identifier of a RO as operands and removes the object from the RO if it doesn't violate certain integrity constraints. The Split operation can be divided into three sorts, i.e., divides a RPO-line into its sub-RPO-line, divides a RPO-plane into its sub-RPO-plane, and divides a RPO-solid into its sub-RPO-solid. The second group of operations support the management of two-way linking between ROs and components of spatial attribute values in the database, such as Register, Unregister and GetRO, etc. Here Register informs a RO roid (i.e., RO identity) to a spatial component scid (i.e., spatial component identity) which is depending upon. Unregister removes such information. GetRO returns the geometry. The third group of operations support the selection of ROs for the construction of spatial object, for example, *Cube*, *Identify*, and so on. Cube returns all ROs together with their roid inside or intersecting a given cube. Identify tries to identify a RO close to the RPO-point given as an operand [12].

Having the RSDD concept, like in Ref.[11], we can define some structures and discuss the relationships between these structures. The difference here is that we must consider the relations between 3D ROs.

3 Spatial Data Manipulation in the Relational Database

Both raster data and vector data related to GIS, the representation of spatial data is quite complex. GIS software vendors have employed a variety of techniques to store spatial data and to link these data with geometry features. All approaches, however, use the concept of a database management system (DBMS) to allow the user to define the specific data element types and formats. A DBMS allows a user to describe the particular contents of a database and the formats of data elements (e.g., integer, decimal, date, character). Although the relational DBMS model for storing attributes is, by far, the most popular approach in the GIS software industry, the relational model is based on the storage of attributes as the standard set of data type (integer, real, string, etc), so the fundamental question is how to represent geometry.

As we discussed above, the problem to manage spatial data and to represent geometry can transmit to manipulate ROs. A RO is composed of primitives such as RPO-points, RPO-lines, RPO-planes and RPO-solids. All primitives can be expressed using points in $N \times N \times N$. So, we can store ROs in relational database, either they are RPO-points, RPO-lines, RPO-planes, or RPO-solids. In other words, we can store spatial points, lines, surfaces and

solids in relational database in a uniform format. Because the RO has a finite expression, we can solve the conflict between the infinite precision real numbers of spatial object and the finite precision number systems of computers.

Once the realm concept has been extended to 3-dimension space, it allows the storage and management of all spatial data (features, attributes, and even raster data) in a relational database. This architecture moves away from the traditional concept of a GIS database in which spatial data is stored in a proprietary graphic format and tabular attributes are stored in a separate database table. The general concept for spatial data in the relational database is presented in Fig.2 by Entity-Relationship model.

Figure 2 describes the database schema. A feature table or view corresponds to a feature class. A feature is an object with geometric attributes[14]. Each feature view contains a number of features represented as rows in the view. Each feature contains a number of geometric attribute values represented as columns in the feature view. Each geometric column in a feature view is associated with a particular geometric view or table that contains geometry instances. The correspondence between the feature instances and the geometry instances shall be accomplished through a foreign key that is stored in the geometry column of the feature table. This foreign key references the GID primary key of the geometry table.

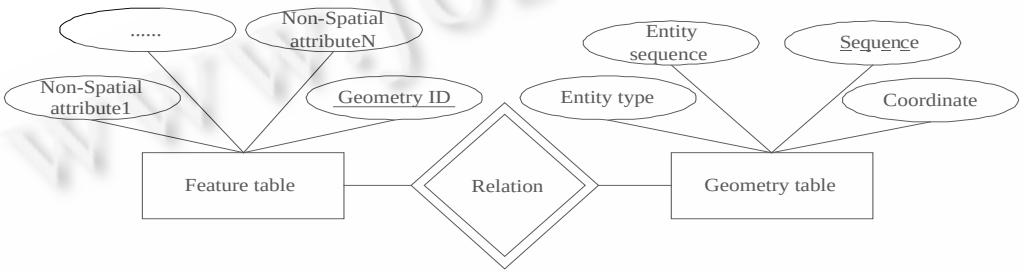
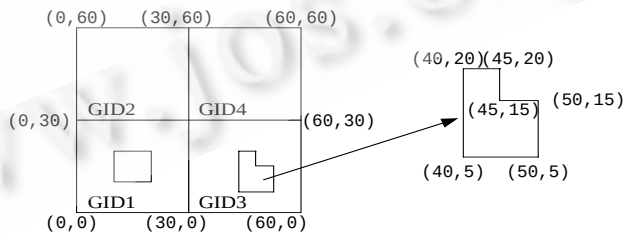


Fig.2 RDBMS-Based spatial data architecture

In the geometry table or view, the normalized geometry implementation defines fixed width tables such as the example in Fig.3[15], the case of 3D is similar. Each primitive element in the geometry is distributed over some number of adjacent rows in the table ordered by a sequence number (SEQ), and identified by a primitive type (ETYPE). Each geometry identified by a key (GID), consists of a collection of elements numbered by an element sequence (ESEQ).



GID	ESEQ	ETYPE	SEQ	X0	Y0	X1	Y1	X2	Y2	X3	Y3	X4	Y4
1	1	3	1	0	0	30	0	30	30	0	30	0	0
1	2	3	1	10	10	20	10	20	20	10	20	10	10
2	1	3	1	0	30	30	30	30	60	0	60	0	30
3	1	3	1	30	0	60	0	60	30	30	30	30	0
3	2	3	1	40	5	50	5	50	15	45	15	45	20
3	2	3	2	45	20	40	20	40	5	Null	Null	Null	Null
4	1	3	1	30	30	60	30	60	60	30	60	30	30

Fig.3 Example of geometry table for polyogon geometry in plane

The rules for geometric entity representation are defined as follows[15]:

- ETYPE designates the geometry type.
- Geometries may have multiple elements. The ESEQ value identifies the individual elements.
- An element may be built up from multiple parts (rows). The rows and their proper sequence are identified by the SEQ value.
- Polygons may contain holes, as described in the geometry object model.
- Polygon rings must close when assembled from an ordered list of parts. The SEQ value designates the part order.
- Coordinates that are not used must be set to Null in complete sets. This is the only way to identify the end of list of coordinates.
- For geometries that continue onto an additional row (as defined by a constant element sequence number or ESEQ) the last point of one row is equal to the first point of the next.
- There is no limit on the number of elements in the geometry, or the number of rows in an element.

4 Conclusion

In this paper, we extend the Markus Schneider's realm concept to 3-dimensional space, thus introduces the RSDD concept. The realm and RSDD concepts solve several problems related to spatial data type for database system[12]. In particular, it solves the problems of numerical robustness, topological correctness and geometric consistency. For spatial data types and spatial databases, this is a satisfactory solution according to generality, rigorous definition, finite resolution and geometric consistency criteria. Our contribution is to manipulate spatial data in 3-dimensional space. Because of the RSDD, we can manage 3D spatial data uniformly in relational database. Thus it provides a method by using relational database to manage spatial data.

References:

[1] Egenhofer, M. Managing spatial storage for structured data. Joint Annual Meeting of the Canadian Cartographic Association and Carto-Quebec. Quebec, May 1987.

[2] Egenhofer, M., Herring, J. High-Level spatial data structures. Geographical Information Systems, 1991,1:227~237.

[3] Egenhofer, M. Managing storage structures for spatial data in databases. Workshop on Geometry and Spatial Information Systems, Fredericton, New Brunswick, July 1987.

[4] Egenhofer, M., Richards, J. Exploratory access to geographic data based on the map-overlay metaphor. Journal of Visual Languages and Computing, 1993,4(2):105~125.

[5] Güting, R.H., Gral. An extensible relational database system for geometric applications. In: Proceedings of the 15th International Conference on Very Large Databases. Amsterdam. The Netherlands: Morgan Kaufmann, 1989. 33~44.

[6] Grumbach, S., Rigaux, P., Scholl, M., *et al*. A spatial constraint database. In: Sophie, C., Richard, H., eds. Database Programming Languages, the 6th International Workshop, Springer, 1997.

[7] Henry, A.S., Korth, F., Sudarshan, S. Database system concepts. 3rd edition. The McGraw-Hill Companies, Inc., 1997.

[8] Burrough, P.A., McDonnell, R.A. Principles of Geographical Information Systems. Oxford University Press, 1998.

[9] Frank, A.U. Properties of geographic data. In: Proceedings of the 2nd Symposium on Large Spatial Databases. Zurich, 1991. In: Lecture Notes in Computer Science, Vol. 525, Springer-Verlag, 1991. 225~234.

[10] Erwig, M. Random access to abstract data types. Technical Report 266, Fern-Universität Hagen, 2000.

[11] Güting, R. H., Schneider, M. Realms: A Foundation for Spatial Data Types in Database Systems. Springer, 1997.

[12] Schneider, M. Spatial Data Types for Database systems. Berlin Heidelberg: Springer-Verlag, 1997.

[13] Open GIS Consortium, Inc. The OpenGIS Abstract Specification. Version 4. 1999.

[14] OpenGIS Consortium, Inc. The OpenGIS abstract specification: an object model for interoperable geoprocessing. Revision 1, 1996. OpenGIS Project Document Number 96-015R1.

[15] Open GIS Consortium, Inc. OpenGIS simple features specification For SQL. Revision 1.1, 1999. OpenGIS Project Document 99~049.

基于关系数据库系统的空间数据处理方法

朱铁稳[1,2], 钟志农[1], 景 宁[1]

[1](国防科学技术大学 电子科学与工程学院,湖南 长沙 410073);

[2](空军第一航空学院,河南 郑州 464000)

摘要: 目前,大量的数据都是通过数据库管理系统(DBMS)进行存储和管理,关系数据库是解决数据处理问题的最成熟和最有效的工具.在地理信息系统(GIS)的应用中,是利用所谓的空间数据库来管理、分析和观察空间数据.因为空间数据包含许多不同的数据格式和结构,这些复杂数据导致了对空间数据的操作和处理是非常复杂和困难的一项工作.提出了一种利用关系数据库的成熟技术来解决空间数据处理的方法,思路是引入 RSDD(regularly spatial discrete domains)概念,并定义基于 RSDD 的基本对象 RPO(SDD_Based primary object)和对象 RO(RSDD_Based object)概念,这些概念能够解决空间对象实数表示的无限精确性和计算机处理的有限精度之间的矛盾.

关键词: 关系数据库;空间数据库;RSDD;GIS

中图法分类号: TP311 文献标识码: A

****

第 4 届软件形式工程方法国际学术会议

征 文 通 知

第 4 届软件形式工程方法国际学术会议 ICFEM2002(The 4th International Conference on Formal Engineering Methods)将计划于 2002 年 10 月 22 日~25 日在上海举行.会议由国家自然科学基金委、上海大学和澳门联合国大学联合主办,中国计算机学会、中国软件行业协会、上海市计算机学会和华东师范大学协办.上海大学承办该次会议.

会议主题包括软件形式方法和其他方法的集成、形式化验证、形式规格说明的确认、基于规格说明的测试、规格说明的演化与求精、形式方法的工具与环境、软件规格说明技术与语言、形式方法的应用、基于形式方法的管理、软件体系结构、组件(Component)工程、需求工程、UML 开发方法、模型检查、形式化语义以及与软件形式方法有关的其他论题.

会议论文集将由 Springer-Verlag Press 正式出版.

澳门联合国大学的 He Jifeng 教授和中国工程院院士、国防科技大学陈火旺教授任本届会议的会议主席(General Chairs),上海大学的缪淮扣教授和澳门联合国大学的 Chris George 教授任本届会议的程序委员会主席(Program Chairs).

论文截止日期: 2002 年 4 月 20; 录用通知发出日期: 2002 年 6 月 20 日

联系地址:200072 上海市延长路 149 号 上海大学计算机学院

联系人:缪淮扣

电话:021-56338101,56337684,65287365

传真:021-56333601

Email: icfem02@mail.shu.edu.cn

有关论文的递交和会议的有关信息可以查询 http://www.shu.edu.cn/icfem2002/index.htm