Presented at the SaGES 2019 Conference, Nicholls State University, August, 2019.

Re-thinking the Data Model: GIS as/is a Relational Database

N.W.J. Hazelton and Yitong Wu

Department of Geospatial Informatics,
Troy University,
Troy, AL, 36082.

Email: nhazelton@troy.edu

Abstract

GIS application software matches spatial data very well with relational database technology for storing attribute data, even if a relational database is not used for storing the spatial components of the data. Early efforts to use a relational database as the foundational storage system for GIS ran into serious performance issues or used databases that were not fully relational (e.g., Empress for the System 9 GIS). More recent systems use extended relational databases as a storage system, in effect acting as a flat-file system of links to external files, but the spatial data component is handled by other software in a non-relational way.

Yet the fundamental nature of GIS, whether raster or vector in nature, satisfies relational logic. GIS operations cannot proceed unless the GIS's spatial data has been normalized to at least the Third Normal Form.

Despite the many advances in making GIS object-oriented, both as they appear to the user and as they operate at a deeper level, whether a database is relational depends solely upon whether or not it can be conceived of by the user as being able to be represented solely using tables that contain straightforward atomic values (i.e., no pointers or similar addressing structures). As GIS commonly have their data perceived as maps and spatial/graphical representations, rather than purely as tables, the question for GIS comes down to whether the user can reasonably represent the underlying data, both spatial and attribute, as simple tables.

Introduction

The earliest developments in GIS were largely built using the grid-cell model. Here the attribute data was contained in separate files or separate structures within a single file, and most systems allowed a single value per grid cell. There were experiments with pixels that contained more than one attribute, termed 'mixcels,' but this was difficult to implement and was not continued.

As grid-cell systems gained greater complexity, there was an effort to move attribute data into some kind of external file. In many cases, this was a simple table, stored in a text or binary format. Sometimes, the table was included in a database, and this was often in dBase III format, as this was popular as a PC-based database management system (DBMS) in the early years of widespread GIS adoption.

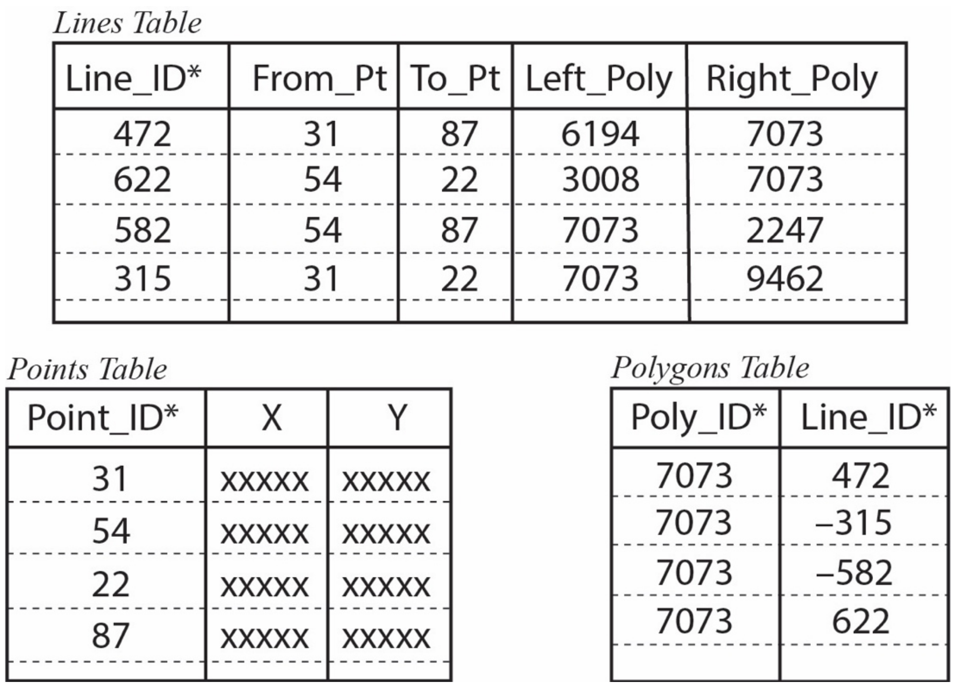
The development of vector-based GIS, such as ESRI's ARC/INFO, required a separation between the spatial data component and the attribute data component, and a relational database was a common means of storing the attribute data. The advantages of using an external database included being able to focus development on the means of storing and processing the spatial data, without having to worry about development of an attribute DBMS.

INFO, the initial database used for ARC/INFO, was an early relational database. In the early days of ARC/INFO, the bulk of database research was focused on relational DBMS (Date, 1988, pp. 19-20). Despite the problems that became apparent in trying to place topologically-related spatial data into a relational database (e.g., van Roessel, 1986), relational DBMS became the attribute database of choice for GIS.

The problems involved in placing topologically-related spatial data into a relational database are concerned with practicality, rather than what is possible. Traditionally, the ARC/INFO topological model for coverage was represented as a series of tables (relations), and while it was certainly possible to store spatial data that way (van Roessel (1986) suggested it as a means of data transfer), practical operation dictated other solutions. The primary problem arises when something as simple as a click to select a data item occurs. To determine which item has been clicked and bring up all the components required to display it involves a massive series of joins of the entire spatial database, and this recurs with almost every operation or query on the spatial data.

As a consequence, ARC/INFO used a form of network database to store the spatial data, while the attribute data were stored in the INFO relational database. In later versions, and into ArcGIS, the spatial data is still moved to a network-type repository for analysis, despite being stored in a range of other data models, such as shapefiles. The spatial extensions to Oracle consist of links stored in relational tables to connect to external network-like data storage.

The object-oriented data model can be adapted to the practical needs of spatial information processing and avoids the need to separate attribute and spatial data so sharply. However, discussion of object-oriented data models in any depth is beyond the scope of this paper.



Lines Table

Line_ID*	From_Pt	To_Pt	Left_Poly	Right_Poly
472	31	87	6194	7073
622	54	22	3008	7073
582	54	87	7073	2247
315	31	22	7073	9462

Points Table

Point_ID*	X	Y
31	xxxxx	xxxxx
54	xxxxx	xxxxx
22	xxxxx	xxxxx
87	xxxxx	xxxxx

Polygons Table

Poly_ID*	Line_ID*
7073	472
7073	–315
7073	–582
7073	622

Figure 1 Classical Points–Lines–Polygons Tables giving a Relational Structure to Vector-Based Spatial Data. The Primary Key in each table is marked with an asterisk. Note that the Polygons table has a composite key. All of the attributes in the Lines table are foreign keys for the other tables. This arrangement of attributes and keys ensures both normalization and avoidance of complex functional dependencies.

Relational Database Definitions

In his efforts to formalize the various definitions relating to relational databases, Date (1988) used somewhat different language to the earlier work of Codd (1970, 1974), to deal with some issues that had become apparent in database design since that time. There have been subsequent refinements, but we will retain Date's definitions.

Relational Database

Date (1988, p. 320) defines a database system as relational "if and only if it supports at least the following:

- Relational databases (i.e., databases that can be perceived by the users as tables, and nothing but tables);
- At least the operations Select, Project, and (natural) Join, without requiring any predefinition of physical access paths to support those operations."

The named operations do not have to be supported explicitly, i.e., they may be invoked by other names and within larger functional processes. SQL is not required to be used as the operational and query language. There is no requirement that tables are indexed (although this is highly desirable for efficient operation), but it must be possible to query on any field, whether it is indexed or not.

Within the spatial data component of GIS, the select operation (retrieving one or more tuples from a table) can be implemented in many ways, such as by running a formal SQL query or (more commonly) by clicking on a visible object. The project operation selects specific attributes (i.e., selects a column from a table), and in GIS that can be achieve by a query concerning all objects that have certain attributes (commonly by using the SQL SELECT statement in a query in a vector-based GIS, but also by threshold modeling in a grid-cell GIS). A join operation in GIS consist of a spatial overlay operation, and is achieved by a number of possible operations, depending upon the nature of the join required and the particular GIS in question.

The user's perception of tables means tables of data in their simplest form. The rows (tuples or records) are unordered, and the columns (fields) are likewise unordered. Each field contains single values (atomic, in that they cannot be divided into anything simpler). Each tuple has a unique identifier field or combination of fields, termed the primary key.

Each tuple in any table is a relation that includes the various fields and can be modeled using the mathematics of relations. The fields in the relations are time-varying, in that values may be changed over time, i.e., the database does not have to be fixed in time but can be a dynamic entity. The relations are normalized to some degree (Date, 1988, p. 245).

The question of the user perceiving the database as tables is important in GIS. A GIS user perceives the data in the GIS as predominantly graphical or spatial and consider it in terms of a map. But as has been argued elsewhere, such as by van Roessel (1986) through to Burrough *et al*., (2015), it is easily possible to consider GIS spatial data in the form of tables, whether a grid-cell or vector-based GIS is under discussion. An example of such a representation for vector-based GIS is shown in Figure 1. Of course, attribute data is readily perceived as tables.

If the spatial component of a vector-based GIS is stored in a network database form, does this invalidate GIS being considered as a relational database? No, the critical issue is the user's perception of the database as tables. This perception is at the conceptual level of the database architecture, not the internal or physical level. Many databases that are sold as RDBMS and can be operated as RDBMS at the conceptual level, are something quite different at the internal level. IBM's DB2 can be considered as a RDBMS at the conceptual level, which is as deep as most users need to go, but at the physical level it is primarily an inverted list type of database. So the relational definition is concerned purely with the conceptual level of the database architecture, and there is no need for a truly relational implementation at the internal or physical level.

Keys

Data is retrieved from a relational database by means of keys. There are a number of different types of keys that need to be considered, and they will be defined here.

A **Candidate Key** is one or more of the attributes (fields) in a relation that provide a unique means of identifying every tuple on the table. No tuples in the relation (table) have the same values for that particular attribute or combination of attributes. Further, no parts of the key can be discarded without eliminating the uniqueness of the key. Each table must have one or more candidate keys.

A **Primary Key** is a candidate key that has been chosen as the sole tuple-level addressing mechanism for a table (relation). The addressing mechanism is usually built into the nature of common queries and other operations for the table. A primary key is guaranteed to return a single tuple (at most) for a given value or set of values supplied for the attribute or attributes that make up the primary key.

For the spatial data in a GIS, the primary means of addressing the data is always location. However, this is often carried through some other unique entity that can be determined using location. So the classical relational model of 2-D vector-based GIS is built around lines (arcs or edges), each of which is unique and can be addressed using its defining points (nodes), which are addressed by their spatial location.

A **Foreign Key** is an attribute value (or combination of values) within a table that allows tuples within that table to be selected, but they value(s) are primary keys in a different table. This means that the foreign key does not need to be unique within the table where it is a foreign key. In vector-based GIS, the best example of this is the foreign key of points (nodes) in the lines (arcs or edges) data table (see Figure 1). Individual points clearly can appear more than once in a lines table, as more than one line can share a common point. So while points act as a foreign key in the lines table, the points identifier is the primary key in the points table.

There are certain integrity rules that must be observed for keys in relational databases. There must be entity integrity, in that no attribute in a primary key can have a null value. This means that all spatial information in a GIS must have a location. Referential integrity is required, and this means that any foreign keys must match a primary key in another table (relation) or be null. Foreign keys are the means by which attribute selections (within the attribute database) are used to select spatial components (within the spatial database).

Dependencies

An important concept in relational database design is functional dependence (FD). This is where a given attribute is 'dependent' on another attribute in all the tuples in a table. Date (1988, p. 365) provides the following definition of functional dependence:

"Given a relation *R*, attribute *Y* of *R* is functionally dependent on attribute *X* of *R* if and only if, whenever two tuples of *R* agree on their *X*-value, they must necessarily agree on their *Y*-value."

In effect, this makes attribute *X* a candidate key for attribute *Y* and is an important requirement for candidate keys. If all attributes in a relation (table) are dependent on attribute *X*, then attribute *X* is a candidate key for the relation. Attributes *X* and *Y* may be composite, i.e., made up of multiple attributes. In Figure 1, the Polygons table has a composite key: Poly_ID and Line_ID form the primary key.

If an attribute *Y* is functionally dependent on attribute *X*, but not functionally dependent on some subset of attribute *X* (i.e., *X* is a composite of several attributes in the relation (table)), then attribute *Y* is fully functionally dependent on attribute *X*.

Transitive dependencies occur when the primary key (attribute *A*) functionally determines attribute *B*, but that attribute *B* determines attribute *C*, which means that attribute *C* has a transitive functional dependence on the primary key, *A*. This situation tends to occur in complex collections of data, but tends not to occur in the spatial part of GIS databases. Looking at the essential table structure of a vector-based GIS in Figure 1, we can see that there are no functional dependencies in any of the tables, apart from attributes that are functionally dependent on the primary key. In a grid-cell GIS, the single-value nature of each grid-cell in each layer means that there are no transitive dependencies.

Transitive dependencies necessarily occur across relations (tables), and the primary key in one relation acts as a foreign key in another relation. This is how tables are linked. However, transitive dependencies within a single relation can make for difficult operations, and are best avoided.

Multivalued dependence (MVD) occurs in certain forms of relations (tables) where there are repeating groups of attribute values. This situation leads to having indeterminate ways of storing data, owing to the odd combinations that can arise. If we have a relation, *R*, with at least three additional attribute values, *A*, *B* and *C*, then a multivalued dependency exists between *A* and *B* if and only if the set of *B*-values matching a give (*A*-value, *C*-value) pair in *R* depends only on the *A*-values and is independent of the *C*-value. The attribute *A* is the key, and *A*, *B* and *C* can be composite (Date, 1988, p. 384).

Normalization

A critical part of relational database design is normalization of the data. While normalization is not a hard-and-fast rule, normalization avoids redundancy and allows more efficient operations on the database. As a general guideline, relational databases need to be normalized to the Third Normal Form (3NF) to ensure that basic operations work properly. Higher normal forms are possible and are important when building relational databases with a number of many-to-many relationship. As this is a condition of spatial data, higher normal forms will need to be considered.

A relational database may be deliberately left unnormalized for a number of reasons, mostly to allow more efficient operation for complex queries. In such cases, the designers need to be aware of the potential for redundancies and potential problems in operation, and build in methods to deal with these, again hiding the non-relational aspects from the user's perception.

Normalization can be thought of as trying to make relations as near as possible to being independent of each other, in the sense that dependencies between tables occur only through appropriate keys. Tables are progressively simplified through higher normal forms as a means of minimizing redundancies that will adversely affect operations. Sixth normal form (6NF) occurs where there is little or no reduction that can be done on the tables in the database to simplify them any further. Note that 6NF is not always necessary, and may not be helpful for a working database.

First Normal Form. A relation (table) is in first normal form (1NF) if and only if all underlying domains (fields) contain atomic values only (Date, 1988, p. 367). This means that for every data item in any position in any table in the database, there is always exactly one data value present. This means that multiple values of an attribute are not permitted.

In a grid-cell GIS, this is very simple to realize. In each layer of the GIS, each cell contains a single value, or if a lattice is envisaged for the layer, each lattice point has a single attribute value. For most GIS applications this works well, unless we try to picture layers as though they were a stack of 2-D maps and then consider modeling a 3-D world. So ‘layers’ need to be considered as a very different way to group spatial regions based on attribute but clearly differentiated from elevation analogues.

For a 2-D vector-based GIS, the process of placing a given layer in 1NF involves what was termed ‘Clean and Build’ in the coverage versions of ARC/INFO. This process ensures that polygons are closed and that any given location within the layer has a single possible attribute value. For shapefiles, similar requirements must be met. This process of meeting 1NF is usually considered to be meeting topological integrity in vector-based GIS.

It is certainly possible to draw a map with spatial anomalies or errors, such as contour lines that cross, unclosed polygons and dangling nodes. These anomalies lead to ambiguity in the value of attributes at any given spatial location and must be removed before display or analysis can be successful. Therefore, working GIS must have their spatial data in 1NF before operations can proceed successfully.

Second Normal Form. A relation (table) is in second normal form (2NF) if and only if it is in 1NF and every non-key attribute is fully dependent on the primary key (Date, 1988, p. 370). For the spatial data in a GIS, the primary key is always location (in its simplest form an X, Y attribute pair). In a grid-cell GIS, the location determines the specific cell, which in turn determines the attribute value, and so each cell’s attribute value is full dependent on the primary key.

In a vector-based GIS that is in 1NF, i.e., has topological integrity in each layer, location devolves through a series of tables, and each table carries single points, lines or polygons, which are related through foreign keys. Each point, line or polygon necessarily defines a single attribute value, which is commonly a key into an attribute table. Enforcing basic topological integrity in a vector-based GIS allows each layer (and so the relevant tables and therefore the GIS) to be in 2NF.

Third Normal Form. Third Normal Form (3NF) evolved through several stages, as it was found necessary to refine the definition to deal with more complex cases. The essential issue with 3NF in its initial form was transitive dependencies. The initial formulation, that every non-key attribute in a table was non-transitively dependent on the primary key, is discussed in the Dependencies sub-section, above. However, this definition had difficulties in certain cases where there were multiple candidate keys for a relation, and the candidate keys were composite and had overlapping attributes (a situation not found in the spatial databases of GIS).

Boyce/Codd Normal Form (BCNF) was devised to deal with situations with multiple candidate keys. In the definition of BCNF, the term ‘determinant’ means any attribute upon which some other attribute if fully functionally dependent. A relation (table) is in Boyce/Codd Normal Form (BCNF) if and only if every determinant is a candidate key (Date, 1988, p. 374). While BCNF was developed over a period of several years, it is equivalent to the form of 3NF given by Heath (1971). BCNF has become the *de facto* 3NF in RDBMS design, even though it is a ‘stronger’ normal form than the original 3NF.

In GIS spatial databases, the determinant in each table is the primary key and all other attributes are fully functionally dependent upon the primary key. This can be seen in Figure 1. As the primary key is the only candidate key, the spatial part of GIS databases is in BCNF, as well as the more general 3NF. Grid-cell GIS are also in BCNF and 3NF.

Fourth Normal Form. Fourth Normal Form (4NF) deals with issues caused by repeating groups of data within tables that would otherwise meet BCNF. The concept of multivalued dependency was introduced to deal with this repeating groups problem, along with 4NF. Date (1988, p. 385) defines 4NF as follows: “A relation *R* is in fourth normal form (4NF) if and only if, whenever there is a multivalued dependency in *R*, say *B* is multivalued dependent on *A*, then all attributes of *R* are also functionally dependent on *A*… Equivalently: *R* is in 4NF if it is in BCNF and all MVDs in *R* are in fact FDs."

In Figure 1, the Points table has a multivalued dependency, however each of the co-ordinate value (X, Y) are functionally dependent on the key (Point_ID), but not on each other. In effect, we can consider (X, Y) to be a composite attribute of the key, as we cannot easily consider co-ordinate pairs as disconnected entities. It is also not realistic to expect the co-ordinates to repeat, other than by chance.

Also in Figure 1, the Lines table has multivalued dependence, however, we can reasonably consider all attributes other than the key to be a single composite attribute. That composite attribute is functionally dependent on the key. Therefore, the tables in Figure 1 are in 4NF.

If repeating groups of attributes are a problem, tables can be decomposed into projections until the repetitions are eliminated. This would take us on the path to 6NF.

In a grid-cell GIS, all attributes are accessed using the primary key, which is the spatial location, as each layer or image is, in effect, a single table, and there are no MVDs in any table, as a single attribute with a FD only on the primary key. Grid-cell GIS layers are therefore in 4NF.

Fifth Normal Form. Fifth Normal Form (5NF) was developed to deal with a specific set of problems. For all normal forms through 4NF it is possible to normalize a table by split it into two projections of the table. These could be recombined by a join to form the original table without loss of data, i.e., projection into two tables was a lossless decomposition of the original table.

It is possible to create tables in 4NF where it is not possible to decompose them into two tables without a loss of data, or the creation of spurious data following a join. However, it is possible to undertake a lossless decomposition of the table into three or more tables, and this leads to the concept of join dependency, where how a table is decomposed can lead to different results when the resulting tables are joined.

Join dependency is the situation where "relation *R* satisfies the join dependency (JD) *(*X, Y, …, Z*) if and only if *R* is equal to the join of its projections on *X, Y, …, Z*, where *X, Y, …, Z* are subsets of the set of attributes of *R*" (Date, 1988, p. 388). In other words, if a lossless decomposition of a table is not possible by projection, then join dependency is not satisfied.

Decomposition of tables in this situation requires careful examination of the nature of the data, especially its semantics, which may require redesign of the entire database to overcome the inability to decompose tables without loss of data. If a table can be decomposed without data loss by projection, i.e., all the join dependencies are related to keys, then it is in 5NF. Date (1988, p. 389) defines 5NF as follows: "A relation *R* is in fifth normal form (5NF)—also called projection-join normal form (PJ/NF)—if and only if every join dependency in *R* is a consequence of the candidate keys of *R*."

Considering the tables in Figure 1, if we consider that the non-key attributes in the Points and Lines tables are actually single composite attributes in each table, then there is no decomposition possible by projection that would not disrupt the attributes, especially the semantics of the data, i.e., the tables are in their simplest form, and therefore are in 5NF. Similarly, grid-cell GIS tables or layers cannot be decomposed any further, and so are at a stage of lossless decomposition with no join dependencies, and so are in 5NF.

Sixth Normal Form. Sixth Normal Form (6NF) was developed by Date *et al*., (2003, 2014) as a means of dealing with complex dependencies, especially in temporal databases, i.e., databases that include time as a central part of the data. It is also used to simplify storage in data warehousing applications. As GIS to date have not incorporated time very well, and especially not as a fundamental part of the data model, this normal form is rarely considered in GIS management. However, as more data with a temporal component is acquired, and the need for spatio-temporal analysis grows, incorporation of temporal databases into the spatial component of GIS will need consideration.

Date and Darwen (1995) introduced the term 'relvar,' meaning a 'relation variable,' i.e., a table, to avoid ambiguity with the use of 'relation' in database literature, where it can mean both a table and the abstract relation of relational algebra. 6NF can be defined as follows: "A relvar R is in sixth normal form (6NF) if and only if every Join Dependency of R is trivial—where a Join Dependency is trivial if and only if one of its components is equal to the pertinent heading in its entirety" (Date et al., 2014, p. 213). In effect, this means breaking down all relations (tables) to their simplest forms, and led to the simple and intuitive definition: a table is in 6NF when each tuple contains the primary key, and at most one other attribute, or "the result is tables that cannot be decomposed any further; in most cases, the tables include the primary key and a single non-key attribute" (Harrington, 2016, p. 125).

The tables in Figure 1 could be decomposed into 6NF by breaking the Lines table into a series of tables built around the primary key and a single attribute. However, this will tend to lower efficiency for regular GIS operations with current systems, and so would not be undertaken in current circumstances. It also would require the decomposition of the single composite attribute in the Lines and Points tables, while the Polygons table is all primary key. Using this interpretation, one could consider the tables in Figure 1 to be in 6NF. Similarly, as a grid-cell GIS has its layers (tables) of spatial data decomposed to the situation where they consist of a single non-key attribute and the primary key, all grid-cell GIS are in 6NF.

Other Normal Forms. With a database in full 6NF, there is no more decomposition possible: the tables are in the absolute simplest forms. Therefore, there is no 7NF. There have been a few variations within the normal forms, but if the numbered normal forms are followed, there are few circumstances where the database will run into serious problems with duplication of data and similar issues.

Some Other Discussions of Normalization

Discussions of normalization of relations in many of the better-known GIS texts have tended to focus on attribute databases, rather than the spatial database. Laurini and Thompson (1992, pp. 390-391, 490-497) discuss normalization for attribute databases and in very basic ways for spatial data, limiting their discussion (pp. 491, 485) to the normalization of a simple polygon relation into the type of model shown in Figure 1. While they discuss the various normal forms, they restrict this discussion to attribute data, rather than extending it explicitly to the spatial data. They do point to non-normalized relations being useful for spatial data, especially with respect to object-oriented GIS, at that time only available with the System 9 GIS (pp. 390-391). Chang (2002, p. 105-106) discusses normalization only in relation to attribute databases, as does DeMers (2003, p. 111). Other GIS texts (e.g., Clarke, 2003) use 'normalization' only to refer to normalizing attribute values to remove statistical bias, and so are referring primarily to grid-cell GIS.

Future Developments in GIS Databases

As attribute data grows more complex, more complex forms for storing it are required. The relational database model allows the storage of progressively more complex information, and so lends itself to storing attribute data. As a simple example, it is possible to use a RDBMS to store data from a borehole, which is linked to a single spatial point, but contains detailed stratigraphic information, in addition to conductivity, density, magnetic, chemical and other information for the core extracted at that borehole. In addition, on-going records concerning water table heights, salinity and pollutant loads can be added to the RDBMS by means of additional tables.

The ever-increasing amount of spatial data that is available for GIS analysis leads to the need for a temporal capability in GIS that is tied to the fundamental nature of GIS data. In its early days, GIS had much more of a 'project' orientation: build a GIS and database to address a fairly specific issue, then move on. GIS then rarely had the characteristics of Land Information Systems (LIS), which were built around change and longevity, and therefore had to address a temporal component. With multi-decade spatial datasets now available, dealing with the time-aspect of this data, and being able to analyze these datasets over time, is becoming more critical. In addition, the new national datums being released in 2022 by NGS will shift reference frames into a 4-D environment, where time tagging of location will become the norm. GIS must develop a temporal component that attached to points, lines and polygons, as well as attributes, and allows temporal operations at the most basic level.

Such a capability can be built into relational databases, but only at the cost of 'extreme' normalization, i.e., 6NF. Further, the nice data model in Figure 1 must be redesigned to allow for a temporal component for all spatial data. This will require a major overhaul of the fundamental data model for GIS, something that has been actively avoided for many years.

Brief mention should be made concerning non-traditional relational databases and GIS. One of the earliest RDBMS was Empress, which had a significant difference compared to conventional RDBMS. Empress allowed unlimited size entries in tables, which is very different to the pre-defined field lengths and variable sizes required in conventional RDBMS. This allowed Empress to be used as the support database for the System 9 GIS, the first commercial object-oriented GIS, as objects (of indefinite size because they are only created at run-time) could be stored and managed within regular RDBMS tables. Developments in temporal GIS may require similar extensions of RDBMS to operate, rather than being decomposed to a full 6NF and treated as a GIS on top of a data warehouse system.

Object-oriented GIS had a relatively brief appearance in two commercial systems, System 9 and Tigris, but object-oriented systems offer significant potential for spatio-temporal GIS development, where spatio-temporal and multi-spatio-temporal data are handled natively by the GIS. This approach was explored and a data model developed (Hazelton, 1992) that allowed topologically-structured multi-spatio-temporal data in a GIS. It may be a wiser move to consider object-oriented GIS development for native multi-spatio-temporal GIS capabilities in the future.

Conclusions

The purpose of this paper was to make the case that vector-based GIS can be viewed as a full relational database, because the spatial data can be interpreted purely as tables, while the attribute data are usually stored in a commercial RDBMS. As any DBMS can be considered to be relational if a user can interpret the database solely as tables, both grid-cell and vector-based GIS meet this requirement. Further, it can be shown that a properly formed topological structure for a vector-based GIS, and all grid-cell GIS, can be considered to be relations (tables) that are in 6NF.

As database design for attribute data in GIS is becoming progressively more important because attribute data is becoming more complex, it is important that students understand normalization processes for RDBMS. Approaching normalization of the attribute database while interpreting the spatial data in the GIS in terms of relations that have to be normalized is one option for connecting the spatial and attribute components in a way that assists learning of other important aspects of database design.

As GIS evolves and changes to handle truly spatio-temporal data in a more capable way, databases will need to be extended, and data warehousing methodologies such as normalization to 6NF will need to be incorporated in into students' skill sets. This paper outlines a methodology for teaching a number of these concepts while remaining close to the core components of GIS.

References

Burrough, P.A., McDonnell, R.A., and Lloyd, C.D., 2015. *Principles of Geographical Information Systems* (3rd ed.). Oxford, UK: Oxford University Press.

Chang, K-T, 2002. *Introduction to Geographic Information Systems*. New York: McGraw-Hill Higher Education.

Clarke, K.C., 2003. *Getting Started with Geographic Information Systems*. Upper Saddle River, NJ: Pearson Education.

Codd, E.F., 1970. A Relational Model for Large Shared Data Banks. *Communications of the ACM*, Vol. 13, No. 6, June, 1970, pp. 377-387.

Codd, E.F., 1974. Recent Investigations into Relational Data Base Systems. *Proceedings of the IFIP Congress 1974, held in Stockholm, Sweden, August 5-10, 1974*, pp. 1017-1021.

Date, C.J., 1988. *An Introduction to Database Systems. Volume I.* 4th edition with corrections. Reading, MA: Addison-Wesley Publishing Company, Inc.

Date, C.J., and Darwen, H., 1995. The Third Manifesto. *ACM SIGMOD Record*, Vol. 24, No. 1, pp. 39-49.

Date, C.J., Darwen, H., and Lorentzos, N.A., 2003. *Temporal Data and the Relational Model: A Detailed Investigation into the Application of Interval and Relation Theory to the Problem of temporal Database Management*. Oxford, UK: Elsevier, Ltd.

Date, C.J., Darwen, H., and Lorentzos, N.A., 2014. *Time and Relational Theory: Temporal Databases in the Relational Model and SQL*. Burlington, MA: Elsevier-Morgan Kaufmann.

DeMers, M.N., 2003. *Fundamentals of Geographic Information Systems*. Hoboken, NJ: John Wiley & Sons, Inc.

Harrington, J.L., 2016. *Relational Database Design and Implementation*. Burlington, MA: Morgan Kaufmann.

Hazelton, N.W.J., 1992. *Integrating Time, Dynamic Modelling and Geographical Information Systems: Development of Four-Dimensional GIS*. PhD Thesis, University of Melbourne.

Heath, I.J., 1971. Unacceptable File Operations in a Relational Database. *Proceedings of the 1971 ACM SIGFIDET Workshop on Data Description, Access and Control*. pp. 19-33.

Laurini, R., and Thompson, D., 1992. *Fundamentals of Spatial Information Systems*. London: Academic Press, Harcourt Brace and Co.

van Roessel, J.W., 1986. Design of a spatial data structure using the relational normal forms. *Proceedings of the 2nd International Symposium on Spatial Data Handling held in Seattle, WA*., pp. 251-272.