# PATHWAYS TO SHARABLE SPATIAL DATABASES

*Geoffrey Dutton*
Harvard Design & Mapping Co., Inc.
80 Prospect Street
Cambridge MA 02139 USA
01-617-354-0100
qtm@cup.portal.com

## Why Johnny Can't Read (spatial data)

Recent advances in data interchange standards for spatial information pay increased attention to differences in data models used to encode such information. This answers a very real need: new spatial information systems continue to emerge into the marketplace, each possessing unique arrays of data models and data structures. As time goes on, users will adopt greater numbers of spatial systems, tailored to particular ends, such as municipal, transportation and resource management, emergency planning, law enforcement, desktop mapping, strategic marketing and real estate applications. Each such system deployed costs money, incurs commitments and populates databases. While digital cartographic data exchange standards such as FIPS 173 (STDS) enable databases to feed one another — perhaps losing information, perhaps not — data remains multiply represented in autonomous archives, with all their overhead, inconsistencies and hassles.

Based on their successes, and given the diverse and decentralized market for spatial information handling, GIS and related technologies seem to have matured to a point where vertical applications are gaining significant market share. In this environment, interoperability is far from the norm, and its prospects are not improving. The best practical solution has been to use *Autocad*™, which in addition to providing a near-universal — if limited — data exchange format, also serves as a data repository for several successful GIS products. But practical as they are, *Autocad* solutions are not appropriate for many applications; GIS vendors have yet to fully accept the imperatives of a client-server world order, as relational database vendors have done, and as personal computer software vendors are starting to do.

Breaking down barriers between spatial databases will require agreements to be reached about many details, including data dictionary standards, descriptions of topology and coordinate spaces, feature encoding schemes, handling of time, and formats for embedded metadata and other conceptual tidbits. This paper discusses several computational paradigms for addressing these and other issues: (1) just-in-time database construction from source files; (2) a universal spatial data repository; (3) a standardized spatial query language; and (4) application programming interfaces to databases. While all of these alternatives are found to have merit, none is seen as sufficient, and each is limited by the complexity and openness of data models it is able or willing to communicate.

Economic and other realities of database proliferation will compel users, researchers and vendors to address how to better share spatial data. Whether suitable, scalable, synoptic standards will surface is uncertain, but few earnest efforts are in evidence. To realize its technical conclusions, this paper advocates building connections between researchers and vendors, bridging applications and data models, and transcending data transfer and metadata debates to build a more universal consensus about access to spatial data. But any of the paths it describes may equally well lead to discovery, amplify applications, stimulate industry and serve users in many ways, if followed deliberately and enthusiastically.

**Just-in-time Database Construction**

Some GIS data layers and features don't change very much. Political units, administrative boundaries, subdivision plats, zoning areas, roadways, river reaches and topography are examples of lines drawn on maps that change very slowly, so their GIS representations rarely need be updated. Other themes, such as land cover, wetlands, coastlines, utility networks and store locations may transform themselves daily, monthly, seasonally or steadily over time. If a theme requires repeated updating, it may be best to rebuild its GIS representation from source data whenever a project calls for it, and to delete or archive the representation once the project ends. As source data is usually stored in public formats (such as TIGER files or Digital Line Graphs), there are few operational obstacles to sharing it among applications in a client-server environment other than the time required to read in source data and structure its information to conform to a system's data model.

Even themes that are relatively stable tend to be duplicated excessively. Redundant copies of political, census and administrative boundaries abound in GIS databases, propagated whenever a new project may need them. Some projects may need versions of themes bearing specific timestamps, while others may require their official or most recent realizations. In any case, it always is important to have access to authoritative source data, or to well-constructed databases incorporating it. If one's GIS makes sharing project data difficult, new projects need to build themes from appropriate primary data, document what they did to achieve this, and purge databases from secondary storage when they cease to be needed. Activity logs should be kept – preferably in a DBMS – as evidence of the spatial databases built, and can reconstruct them if necessary. Commercial and academic GIS tools are now available that track database lineage and other data quality parameters, as most GISs still do not manage metadata very well. Such tools enable users to clean out entire directories of intermediate results, saving only source data and beneficial output; the rest tends to be less important and can be reformulated if ever it is truly needed.

On-demand database construction is particularly appropriate when source data comes in public formats such as DLG, SDTS and TIGER, and needs to be accessed by various applications. For some of them, just-in-time may not be a preferred strategy; it may be the *only* approach capable of assuring data integrity. Utilities and local governments using AM/FM and GIS in tandem are particularly vulnerable to GIS data obsolescence; in many instances urban infrastructure data is assembled and maintained in CADD environments, then converted into GIS data for thematic mapping, spatial analysis or decision support. Unless a GIS can access primary AM/FM data directly, its themes are in danger of getting out of date and out of synch (schools that have closed, streets without sewers). Always building fresh GIS databases (or at least whenever any sources have changed) is the most reliable way to maintain their logical consistency, although it may not be the least-work way. However, as is discussed next, using a common AM/FM/GIS database is an alternative solution that might avoid much copying and reconstruction, and assures that GIS applications have access to the most current data available.

**Universal Spatial Data Repository**

As using CADD databases to support GIS activity has proven to be a viable

technology in many milieus, one could argue that a working model for a universal spatial database exists, and it is *Autocad*. In addition to database standards, Autodesk (Sausalito, CA) has provided capable drawing tools and an extensible programming environment that promotes AM/FM/GIS interoperability. Several commercial GISs are rooted in this environment, but also incorporate a DBMS to store topology, feature definitions, attribute data, metadata and their relations (Autodesk is rumored to be teaming with several other vendors to market their own GIS). A large complement of utilities that add GIS value to *Autocad* are also available from third parties. But if *Autocad* is the answer, what is the question?

The main question posed by this paper is *how can any GIS gain access to the highest-quality spatial data in its environment*. This includes software that is not privy to *Autocad* databases, either because it is unable to subscribe to them or none exist to access. While the computer industry has seen many proprietary protocols become *de facto* standards (for example, Hayes modem commands, HP *HPGL*™, Adobe *PostScript*™ and MS *Windows*™), this market-driven process tends to exclude users who don't have the necessary licenses, and can't guarantee that all their applications will be able to communicate effectively in any given setting. Yet even though they are proprietary, such arrangements point toward a better way of sharing spatial data.

Public and private institutions have to maintain corporate databases, which increasingly tend to have spatial components. Ratepayers and customers have addresses; departments and distributors have territories; citizens and consumers have postal codes and enumeration districts. Such facts can be normalized into relational fields and stored in standard tabular database management systems (DBMS). Locational facts and relationships used to spatially index records may be derived by GIS analysis, but it often makes sense to evaluate spatial queries in batches and store the results, rather than attempting to answer them interactively. The worth of this strategy depends on whether the convenience value of spatial indexes exceeds the costs of computing, storing and accessing them. Only an application can answer this for sure, and only for itself.

Making organized geographic facts available to client applications may best be handled by relational and object-oriented DBMSs. Many spatial entities and relationships can be described by querying objects, relations and views incorpo-

rating proximity, connectivity, set membership and other geometric and topological criteria. Much of this information is static, but updates and analyses can cause it to change. Only when new spatial relationships must be built need GIS muscles be flexed; most transactions involve queries that don't refer to coordinate data, and can be handled by aspatial DBMS engines.

Relying on DBMSs to handle GIS queries can work well, but isn't sufficient when users want to generalize or otherwise tailor map data or to explore map space (for example, finding features within buffered areas or querying proximal locations). Such pursuits require interactive access to graphic data and spatial operators, as well as to results returned from *ad hoc* queries. Still, many modeling applications (such as network routing/allocation problems and site suitability studies) can operate entirely with DBMS tabular data, needing graphic data only to display their results. This leads one to conclude that while access to graphics tends to be required only occasionally, the need is highly application-dependent, but when it exists, it may be strong.

If one's application is cartographic, tabular databases aren't much of a help. If not in Autodesk's orbit, one is probably stuck with databases built by the application's vendor; those that aren't have to be imported. There is some hope, however. As the result of military initiatives in "MC&G" (mapping, charting and geodesy), a new standard (a family of them, actually) for cartographic data communication has emerged, called DIGEST. Various implementations of it exist (Vector Product Format — VPF — is the most well-known), as well as one data product, the Digital Chart of the World (DCW), available on CD-ROM from USGS and other sources. DIGEST isn't a spatial data transfer standard; instead, it encapsulates complete, self-descriptive cartographic databases, ready for online prime time.

DIGEST can handle feature-oriented as well as layer-oriented map data. It can build points, lines and polygons into complex features and describe by degrees topological relations among them. Any data element can have any number of thematic attributes, but only those defined in its database's data dictionaries. There are places to put metadata as well as facts, organized into directories and files. Lots of files (with extensions like .ESR, .EDX, .FSI, .FAC, .END and .TXT), most of them tables. The reason there are so many file types is that many

of them are optional, only showing up when the datatypes they describe are defined: sometimes there are no point features or no text, or maybe topological relations aren't established. Omitting unneeded files is more efficient than storing empty fields, but makes applications work harder to understand a data model and parse its information.

Even a brief tour of DIGEST is well beyond the scope of this work. It is too complex and polymorphic to assimilate in any one sitting, and may be impossible to savor without the aid of software to collate and present facts and relations defined in any given realization (officially called a *profile*), such as DCW. That product is in fact distributed with software to browse, extract and view global map data, and would be virtually useless without it. But at this time, few GIS vendors have announced that they will support DIGEST, and none of them have promised to implement it as a native database mechanism. This should be neither surprising nor disappointing; until enough experience has been gained with DIGEST to validate its viability, it should not be naively embraced. More proven alternatives exist.

**Standardized Spatial Query Language**

Heterogeneous database environments are now very common, in large part thanks to the capacity to link them together via fourth-generation languages (4GL) for querying and updating databases. SQL and other declarative, non-procedural language interpreters permit data to be stored in private, proprietary formats while enabling access to it by any privileged user, locally or over a network. 4GL strategies are now widely used to integrate tabular data belonging to an enterprise, but users of geographic data have yet to be similarly blessed. Although many researchers and some vendors have developed spatial query languages (including extensions to SQL), there has been little progress in industry toward standardizing spatial query protocols.

It is not difficult to imagine the adoption of a set of operators that would enable spatial queries to be incorporated into an SQL *select* command. This basically means standardizing the semantics of a number of adjectives, such as *within, closer than, farther than, adjacent to* and *connected to* (there aren't a lot of them), and making provisions for parameters to qualify them. One would assume that if GIS vendors were strongly interested in interoperability of their databases, they

would have agreed on such protocols by now. But most vendors still lock up their customers' data in software safes from which data can be removed only by translation into interchange formats. They continue to purvey captive databases, although they increasingly recognize and accede to customer demands to integrate existing spatial data into the environments they engineer. Whatever data may exist in a customer's files, GIS vendors are delighted to install it in or link it to their databases. A flurry of custom conversion program creation usually ensues as data models are extended to handle new spatial constructs. This may well do the trick, but it begs the question, entails extra expense for customers, and yields *ad hoc* solutions that must be re-implemented over and over again.

One is understandably tempted to believe that GIS vendors are reluctant to make it too easy for users to access each other's databases using a client-server model. However, the reasons may not all be based on competitive advantage; while spatial primitives can certainly be accessed via standardized queries, it is genuinely difficult to communicate complete data models this way. How a layer-based GIS encodes geographic entities may be very different than how a feature-based one does. When the latitude users have in modeling data is added to this, it isn't hard to understand how SQL and proposed extensions to it might fail to express important aspects of spatial structure. While there are not a lot of spatial primitives that GISs tend to use, they are defined and glued together in systematic and arbitrary ways into objects, features, themes and layers. Unlike SQL, there is no public data language that can transparently access and relate spatial data and models held in proprietary GIS databases, nor is one likely to emerge for a number of years.

**Application Programming Interfaces**

Much what is and isn't going on in GIS data integration can be appreciated by considering technological forces in the PC software arena. Almost suddenly, after a decade or so of development, PC environments appear to be on the cutting edge of information processing technology. In many respects, the data-handling capabilities that personal computer users now or soon will have at their command rival those of workstation software, including GIS. Much of this power devolves from a rapid maturation of software architecture for handling complex documents.

There is an unmistakable trend in personal computer software engineering to provide increasingly open interfaces between applications in order to exchange "live" data. Many of these schemes, such as Microsoft's *OLE* (object linking environment) and Apple's *Amber* project, are object-oriented and describe protocols for integrating diverse datatypes such as word-processing documents, images, drawings, spreadsheets, video and sound. It is indicative that some Macintosh "works" programs integrate their component files via Apple's publish-and-subscribe system protocols, rather than via internal mechanisms such as calls, global variables and common databases. This style of architecture depends heavily on codification of an application programming interface (API) which standardizes protocols for accessing data from different applications and even for controlling them.

APIs aren't restricted to mediating interactions within and between compiled applications. With suitable utility software, the ability to generate, route and interpret inter-application requests can be placed directly in the hands of users. One way to achieve this is by empowering online queries to servers. SQL is an API to data which is usually thought of as a 4GL instead because it is uttered by users rather than software (although software may also formulate and issue SQL queries and directives). Although it is very useful, SQL by itself is not an adequate vehicle for communicating properties and applications of spatial data. To be effective, GIS data interchange must take place at both a lower architectural level and a higher conceptual level.

There is nothing mysterious about APIs. All software that has linked objects, functions or subroutines has at least one. Every call to a system or application library obeys the rules of some API, although the rules may change from system to system, application to application and language to language. So, while it takes a lot of work to develop a public, standard API, the task is not at all foreign to software engineers. Still, it does not seem to be easy to define an API satisfactory to all parties in the standardization process. Sadly, GIS vendors have been unable to agree on more than a few, and most of those are *de facto* industry standard APIs like Autodesk's DXF/DWG rather than collaborative initiatives.

The traditional role of APIs is to regularize access not to databases but to software. While the software may include database access functions, vendors of

GISs have been reluctant to publicize their own beyond a circle of trusted application developers. To fill this void, an increasing number of programming tools for accessing spatial data have appeared. They include software from Geodessy (Calgary, Alberta), MapInfo (Troy, NY), and TerraLogics (Lowell, MA). While each of these provides an impressive array of display, query and analysis functions, all depend upon proprietary, black-box database architectures that are not meant to be shared by GISs or other spatial information or mapping systems.

As one joke has it, the great thing about standards is there are so many to choose from. The same goes for APIs: interfaces for network data sharing; interfaces for database access; interfaces for graphic interchange; interfaces for GUIs... all are useful, but too many must be obeyed when designing applications. While incorporating APIs may yield more robust software, it is no easier to build and maintain than were pre-API standalone packages. Like other work, programming seems to expand to fill the time available.

Recently, a commercial paradigm surfaced that tries to deal with apibabble. Oracle Corporation (Redwood Shores, CA) announced software called Oracle *Glue*, intended to serve as a "universal API." Although *Glue* does not directly address specific issues of sharing spatial data, it is advertised as capable of handling all the low-level details of sharing databases among diverse applications in a networked environment of heterogeneous platforms. Oracle likes to call *Glue* "middleware" which is adaptable to a range of data services, portable to MS Windows, Macintosh, Unix and pen-based OSs, hardware-scalable, network-independent and capable of being accessed via 4GL commands as well as through 3GL programming interfaces. Even though it takes high ground in the API wars, this approach may fail to prevail. If all and sundry vendors are required to license it from Oracle, *Glue* may not stick to enough computing surfaces to be a useful adhesive. Even so, this approach has great merit, and points a possible way out of the abrasive nexus that holds GIS databases captive.

**Conclusions**

The art and science of spatial data handling faces a looming crisis in database access. Even though more than ten years have elapsed since GIS first walked out of the lab onto the street, and despite the number of systems that have since emerged, users and their databases tend to exist more as populations than as

communities, many islands of automation that only occasionally signal one other. A few simply reject direct questions, others can't answer them, and the rest prescribe what one can ask, and sometimes how to ask for it.

Most graphic data interchange standards are an evasion of the problem, not a solution. Think of how many have come and gone: SYMAP packages; CalComp plotfiles; Tektronix bytestreams; GKS metafiles; GINI and IGES; SIF and DXF; VPF and SDTS. Some had their day and deserved to die, others remain useful, but few offer online access to foreign data. When one compares this state of interoperability to that of tabular databases, one realizes how little has been accomplished and how much more data, currently inaccessible, may be at stake.

As this paper has tried to recount, there is clear evidence that spatial databases are on their way to opening their architectures. Although GIS and mapping system vendors haven't tended to be in the forefront of this movement, they should not be blamed for all the problems that need to be solved; spatial data is complex, heterogeneous, bulky and often application-specific. In addition, not every spatial database needs to be shared, nor is everything it contains of equal value to most users. But the pace at which digital spatial databases are accumulating is now so great that a real urgency exists for finding better ways to share them, and everyone who sells, builds and uses them must work together to broaden the pathways that connect them.