Introduction

THE EDITORS

Like many other technologies, digital computers have evolved to provide ever more sophisticated environments for their users. Early programmers worked with very simple languages that in principle could do anything, but in practice were limited by the complexity of the necessary programming. Today's programming languages, and application programmer interfaces, allow far more to be achieved with much less effort. While it may have taken a million lines of code to write an early GIS in the 1960s or 1970s, the same could probably have been achieved with at least two orders of magnitude less, had it been possible to take advantage of the sophisticated programming environments available today. Languages like Tcl/Tk, for example, allow easy-to-use graphic interfaces to be constructed quickly that would have taken vastly more programmer effort 20 years ago.

Such progress relies on a simple principle: that if enough commonality can be identified between the needs of a sufficiently large number of users, then it makes sense to embed those common needs in the computing environment. Like the human mind, the digital computer is capable of supporting ever more complex concepts provided they can be constructed from simpler ones (and ultimately a 'hard-wired' base) in well-defined ways. Besides obvious gains in efficiency and productivity, such approaches provide consistency and rigour, offer simplicity by hiding the complex workings of operations from the programmer or user, and allow for uniform approaches to such issues as integrity.

By the mid 1960s, the computer industry had begun to see how this principle might be applied to the datasets processed by digital computers. Computer applications had been growing rapidly in various areas of industry and commerce, and were requiring and producing increasingly complex masses of data. Rather than treat each application as unique, and program its operations from scratch, there appeared to be sufficient commonality in the ways these applications interacted with data to justify the development of generic structures and approaches. Thus the database industry was born, in the form of special software applications to manage the interactions between programs and data. By assigning standard data management operations to generic systems, these so-called database management systems (DBMS) relieved the programmer of much inherently repetitive programming. They also encouraged a more disciplined approach to data management, which was perceived to have its own benefits in terms of increased efficiency and control.

While the database industry is by definition generic, and the characteristics of geographical data and GIS widely acknowledged to be special in many respects, nevertheless by the late 1970s significant efforts were under way to take advantage of database technology in GIS applications. Instead of a monolithic, stand-alone software application, GIS was increasingly perceived as layered, with specialised software working in conjunction with, or conceptually on top of, a standard DBMS. ESRI's ARC/INFO was one of the first of these, released in 1981 and incorporating an existing DBMS into a specialised GIS environment. Today, more and more of the functionality of GIS is assigned to increasingly sophisticated but still generic database products, many of which now include the capability to store and process explicitly spatial data.

These moves towards reliance on underlying DBMS reflect several important priorities and concerns in the GIS industry. First, if GIS and underlying DBMS are at least partially independent, then one DBMS can be easily replaced with another. This is attractive to many GIS customers, who may be able to share the DBMS among many computing applications within the organisation, and value the freedom to update the DBMS independently of the GIS. Second, the DBMS may be perceived as more reliable than less generic approaches to data management, because of the relative size of the DBMS industry – an industry more sophisticated in its

approach to data management, with better ways of ensuring data integrity; offering greater interoperability between software environments; and with greater adherence to general standards.

Michael Worboys begins this section with a discussion of database models (Chapter 26). The first generations of database systems, appearing in the 1960s, were regarded as too general for effective use in GIS, and it was not until the emergence of the relational model, with its greater sophistication, that GIS began to adopt database solutions in earnest. The term 'georelational' is often used to describe the particular implementation of the relational model for geographical data, in which geographical relationships between entities become the basis for many of the common keys or linkages between relational tables. Nevertheless, this idea took some time to emerge, and early uses of relational databases in GIS were driven largely by the more general advantages of database systems listed earlier.

Worboys takes the reader beyond the relational model into more recent research and thinking in database systems for GIS, notably the concepts broadly known as 'object-orientation'. Just as the relational model gave GIS users a natural way to represent geographical relationships, object-oriented models provide a natural way to manipulate the various entities found on the geographical landscape, and to describe their behaviours. As Worboys notes, object-oriented databases are in their infancy, and although several successful object-oriented GIS have appeared in recent years, there is still much work to be done in identifying the exact limits of the application of object-oriented thinking in GIS.

The designer of a generic solution to management of data must make decisions based on expectations about usage that will inevitably reflect the needs of the largest segment of users. As a specialised application and a relatively small part of the DBMS market, GIS has its own particular needs that are often difficult to promote in the wider arena of DBMS design. GIS databases tend to be large (a single remotely-sensed image or topographic map can easily require 100 million bytes of storage); and searching for geographical objects based on their locations is inherently multidimensional. DBMS solutions for GIS have often encountered disastrously poor performance, even though it is often possible to 'tune' a modern DBMS for the particular characteristics of a given application. Early users of relational DBMS for GIS found it necessary to develop complex implementation guidelines to ensure minimally acceptable performance. Unfortunately, it is almost always true that the benefits of generic solutions must be balanced against the inability to optimise a generic design for the specific needs of a complex application.

Spatial indexing offers one of the most powerful tools to affect and improve performance in a GIS application, just as indexing in publishing or library cataloguing affects the usefulness of those fields. In the second chapter of this section, Peter van Oosterom reviews the state of the indexing art in spatial databases. Many indexing schemes have been devised, and it seems unlikely that any one is optimal over any significant domain of GIS applications. Many different schemes have been implemented, but although spatial indexing is often invisible to the user, it seems likely that in those applications where performance is critical, some degree of involvement of the user in the implementation of indexing will always be necessary.

Early DBMS followed one or other of the standard models for databases, but used proprietary languages for interaction with the user. Even though the underlying structure was essentially the same, a user wanting to move from one DBMS product to another often had to learn an entirely new language. The introduction of standard query languages, notably SQL, across entire sections of the DBMS industry led to much greater interoperability between systems, and greatly reduced the complications of training users. Recent efforts to extend SQL to the needs of GIS are reviewed by Worboys, while Max Egenhofer and Werner Kuhn in Chapter 28 give an overview of user interaction in general, comparing the query language approach to other, newer, and more powerful methods of user interface design. As an inherently visual technology, GIS stands to benefit enormously from graphic user interfaces, which offer the potential to make GIS much easier to use, and much easier to learn. Egenhofer and Kuhn review the various metaphors that are guiding contemporary user interface design for GIS, and that make use of an increasing number of distinct media.

In the final chapter in this section, Yvan Bédard adds a distinctly practical flavour to the topics discussed in the previous three. While databases provide the broad framework for describing the geographical world, the specific details of implementation can be critical, in determining performance, and essential to the success of any given application. Generic tools have been developed for database design, and much effort has gone into adapting these to the special needs of GIS.