DEVELOPMENT OF A GEOGRAPHIC DATA MODEL FOR HYDROLOGICAL MODELLING

M J Bollaert
200272664

Submitted in partial fulfilment of the
requirements for the degree of MSc in Hydrology

School of Bioresources Engineering and Environmental Hydrology
University of KwaZulu-Natal
Pietermaritzburg

November 2006

PREFACE

The work described in this dissertation was carried out in the School of Bioresources Engineering and Environmental Hydrology, University of KwaZulu-Natal, Pietermaritzburg, from January 2004 to December 2006, under the supervision of Mr. David Clark.

These studies represent original work by the author and have not otherwise been submitted in any form for any degree or diploma to any university. Where use has been made of the work of others it is duly acknowledged in the text.

Signed: …… Date: ……

M.J.Bollaert (author)

Signed: …… Date: ……

D.J.Clark (supervisor)

ACKNOWLEDGMENTS

I wish to express my sincere appreciation and gratitude to the following for the assistance they rendered during this study:

- Mr David Clark, for supervising this project and providing invaluable assistance throughout the duration of this study;

- Professor Roland Schulze, for his assistance and mentorship during my studies;

- Mr Mark Horan and Mr Sean Thornton-Dibb, for the time spent assisting with technical matters;

- The School of Bioresources Engineering and Environmental Hydrology, for providing a working environment and resources making this study possible;

- Mr. Thomas Bech from the Danish Hydrological Institute for his assistance with their dfs0 format;

- My friends, for their help with various problems I encountered;

- My family, for the many years of encouragement and support they provided; and

- My Heavenly Father, Jesus Christ, for the many blessings he has given me, including the opportunity to undertake this study.

ABSTRACT

Hydrology is a multi-disciplinary science, and therefore derives data from diverse sources, with the data often of a spatio-temporal nature. A recent trend has been to combine these data with GIS, due to the data's geographic origin, and inherently requires an abstraction of reality in order to deal with the multitude of data that would otherwise result. Consequently, data models have been developed for this purpose, and these require a generalisation of processes and variables, while offering a simplified structure for their storage.

The purpose of this study was to develop a data model for the storage and dissemination of hydrological variables and associated data used in hydrological modelling. Data would be of a spatial and temporal nature, and thus the design of the new data model needed to provide for this. A number of existing geographic data models were therefore reviewed, including the geodatabase model. This data model and the object-relational database model upon which it was built, were ascertained as being the most suitable for the study, and were therefore included in the design of the new data model. The related Arc Hydro data model was subsequently reviewed, since it offered an established means by which to model geographic features associated with surface hydrology. Following this, an investigation into time series storage methods was carried out, as it was important that the new data model be able to store large time series datasets in an efficient manner. Thus a number of methods were identified and evaluated as to their advantages and disadvantages.

A new data model was thereby conceived, using the geodatabase as its foundation, and was developed in order to offer efficient storage of hydrological data. The data model developed was subsequently tested by populating it with data from the Quaternary Catchments database which supports the ACRU model. Finally, additional functionality was added to the data model, in the form of export options.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY OF TERMS

Attribute data	Tabular data describing the geographic characteristics of features
Data model	An abstraction of the real world which incorporates only those properties relevant to the application at hand
Data structure	The logical arrangement of data as represented by in computer form
Data type	The storage format of data within a database as illustrated by text and numbers
Entity	A real world object that cannot be further subdivided into similar objects. In a relational database, it is an object and its related attributes
Feature	A representation of a real world object on a map
Feature class	A collection of geographic features with the same geometry type, attributes, and the same spatial reference
Feature dataset	Collection of feature classes stored together that share the same coordinate system
Feature type	A grouping of similar features such as roads and rivers
Layer	A visual representation of a geographic dataset in any digital map environment, usually divided according to feature type
Object	In GIS, a digital representation of a non-spatial entity
Object class	A collection of non-spatial data of the same type or class
Representation	A method of illustrating data so it can be viewed and understood
Spatial data	Information about the locations and shapes of geographic features and the relationships between them, usually stored as coordinates and topology
Spatio-temporal	Relating to both space and time
Topology	The relationship of connectivity, or adjacency, which a feature has relative to other features

1. INTRODUCTION

Hydrology may be defined as the science dealing with the properties, distribution, and circulation of water on the surface of the land, in the soil and underlying rocks and in the atmosphere (Maidment, 2002). This concise description alludes to the multi-disciplinary nature of the science, and the many variables and processes it tries to model or observe. As a result, there is a substantial volume of hydrological data and related information, which has been recorded and generated in an attempt to better understand the science. Many of these data are spatio-temporal in nature, and have been integrated with other applications such as hydrological models, in order to perform geographic and scientific analyses, as is evident through the use of Geographical Information Systems (GIS) in the discipline. This integration of hydrology with GIS requires an abstraction of reality in order to deal with the multitude of data that would otherwise result from attempting to record real world hydrology in its full intricacy. Consequently, numerous data models have been developed in an attempt to provide a structure to store data describing hydrological processes and their variables in a simplified approach.

The purpose of the study described in the chapters which follow, was to research and then develop a data model which could deal with the many hydrological variables and their associated data that exist either spatially or temporally, and thereby create a framework for their storage and dissemination. Accordingly, a number of existing data models were reviewed, which primarily dealt with the storage of geographic data. These data models were predominantly sourced from the Environmental Systems Research Institute (ESRI) which has been at the forefront of GIS development, and consequently their data models remain both ground-breaking and popular. Data models are built upon database models, and research into the relational and object-oriented database models was therefore required. This included researching the more recent object-relational approach, which combines benefits found in the relational and object-oriented models. Subsequently, the relatively new geodatabase data model, which combines concepts found in the geographic data models and database models, was assessed. The review of the geodatabase was due to its inclusion into the study as the foundation upon which the proposed hydrological data model would be built. The Arc Hydro data model was also evaluated for its potential to describe the spatial context of natural water-related systems.

The development of the new data model focused primarily on the storage of attribute data related to hydrological modelling, and was directly aimed at supporting the ACRU model, and its associated datasets, while remaining generic enough to provide the potential for interfacing with other hydrological datasets and models. As a test of the suitability of the new data model it was populated with the Quaternary Catchments Database (QCDB) containing spatial and temporal data for use in the ACRU model. Since temporal data form a significant portion of collected data and modelled hydrological values, an investigation was conducted into various time series storage methods and their compatibilities with the geodatabase data model, with a focus on voluminous time series datasets, and their storage in an efficient manner. The selection of a time series storage method for implementation in the remainder of the project influenced the design of the new data model. The time series storage method selected also needed to be suitable for storing rainfall and temperature time series data as part of the QCDB.

The design of the new data model was, above all, required to be generic and extensible. Generic in the sense that it would be suitable for the storage of data types for use by any hydrological model, and extensible such that the inclusion of new hydrological variables would not require changes to the structure of the data model. Additional requirements of the new data model were to:

- Store voluminous amounts of time series data;
- Store attributes for spatial features and non-spatial objects;
- Be able to include both input and output variables of the related hydrological models;
- Store a range of data types suitable for all hydrological models;
- Store units of measure for attributes where relevant; and
- Minimise the amount of data replication to maximise database storage efficiency.

These conditions consequently formed the premise of the data model and its structure. The data model also needed to be compatible with the geodatabase, as it would serve as the GIS and relational framework for its implementation. Arc Hydro was used in the study, since it provided the new data model with an established hydrologically related spatial component. A simplified structure of this dissertation is displayed in Figure 1.1.

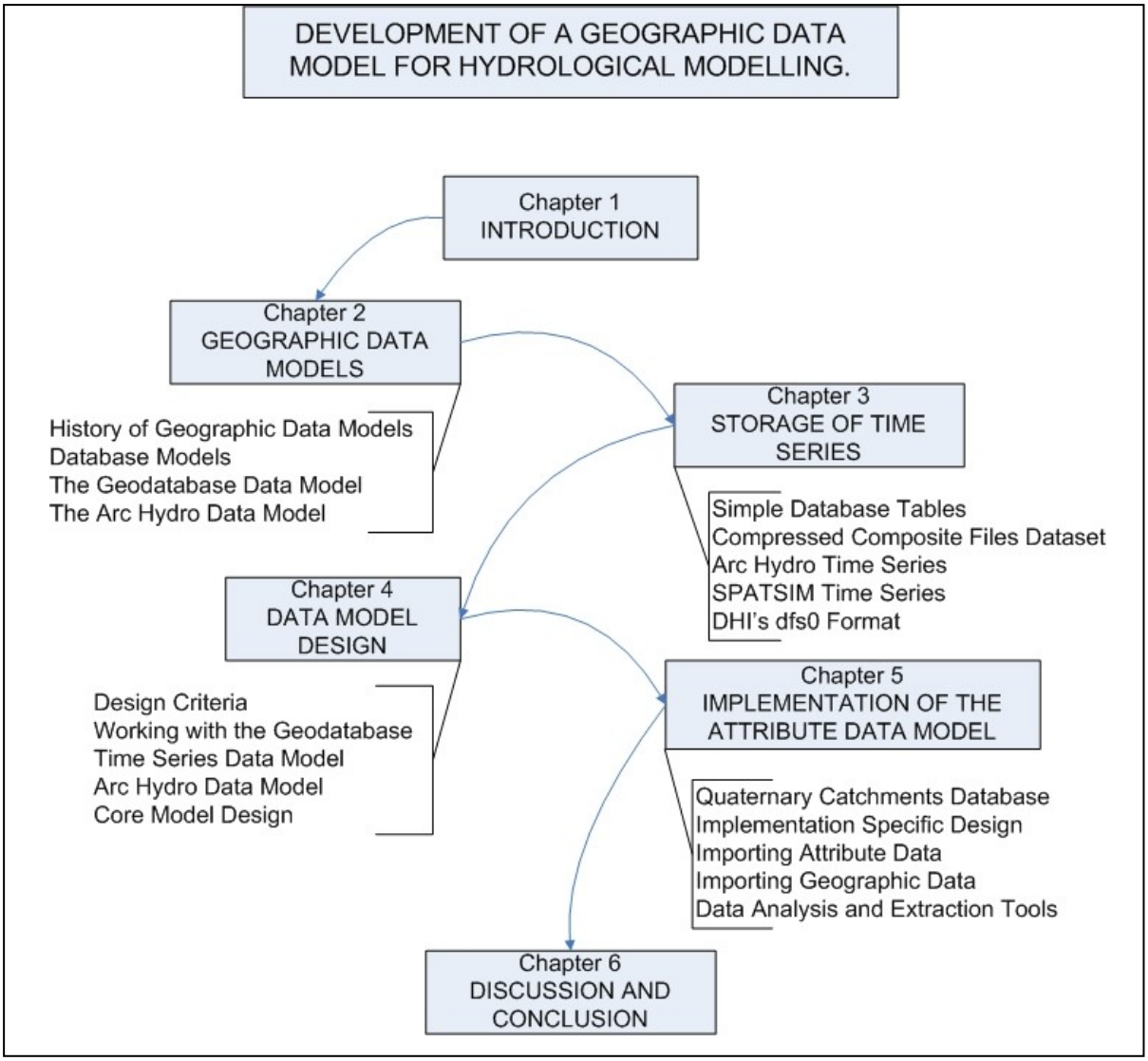


Figure 1.1 A guide to the structure of this dissertation

2. GEOGRAPHIC DATA MODELS

There are many ways by which to model objects, or systems, of the real world. Rivers may be modelled either as a network of lines, a border between two areas, as a local feature with braids and banks, or as a winding line forming a trough in a surface model (Maidment, 2002). The Association for Geographic Information (AGI) defines a data model as an abstraction of the real world, where the data model incorporates only those properties relevant to the application at hand. A data model would, therefore, normally define specific groups of entities, and their attributes, as well as the relationships between entities (AGI, 2006). Thus the purpose of a data model is to provide a simplified description of reality to the desired level of precision, thereby helping to understand the static and dynamic properties of the world. As a result, the data model is at the core of any information system. A data model can thereby be evaluated as to the efficacy with which the information significant to the user is emphasised, and details that are insignificant to the user are suppressed (Raza, 2001). In GIS, there are a number of geographic data models available, where the context of the problems to be solved will guide which model is best for a particular scenario.

In 2000 ESRI introduced a new object-oriented data model, *viz*. the geodatabase data model (Nabar and Patel, 2003). This model has the ability to represent the natural behaviours and relationships of a feature. In order to understand the significance of this new data model, the two primary predecessors to the geodatabase, *viz.* the computer-aided design (CAD) data model and the coverage data model are reviewed. Although not as ground-breaking as the two aforementioned data models, the shapefile data model will also be reviewed, as it has since become widely used. These data models will be reviewed as they were the precursors leading to the introduction of the geodatabase – the platform on which the hydrological data model outlined in this project was implemented. The geodatabase was chosen since at the time of writing, it was the only widespread geographic data model that was implemented in a Relational Database Management System, thereby enabling the development of a new data model within a relational framework.

The geodatabase enables features to be characterised more naturally by allowing for the definition of types of objects and relationships and by capturing how these objects interact with one another (Zeiler, 1999). Furthermore, it allows for the storage of four representations

of geographic data – vector, raster, triangulated irregular networks (TINs), and addresses and locators – making it a unified data model. These representations of geographic data, as well as other benefits of the geodatabase, will be detailed in subsequent sections of this chapter. Apart from this ability to cope with the common spatial representations of geographic data, the geodatabase also includes the functionality of the database models upon which it is built. This chapter will also review the relational and object-oriented database models.

The design of a geodatabase is an important component in its implementation, as the geodatabase developed must ultimately meet the requirements of its users. Development of industry-specific data models, built upon the geodatabase, accordingly allude to its value, as is evident in the success of the Arc Hydro data model (Maidment, 2002).

2.1 History of Geographic Data Models

DeMers (2000) describes a map as simply being a model of spatial phenomena. Thus it is not the role of a map to show every detail, but rather to provide a simplification of reality, relevant to its particular purpose. Although maps have been created and read throughout much of human history, prior to the twentieth century they were only available as printed documents that used sheets of paper or parchments as a means to record geographic information. This information would have been in the form of objects and features such as routes, cities, and coastlines. The advent and progress of computers, coupled with the development of GIS technology, allowed for the creation of maps in either a printed or digital form since as early as the 1960s (Coppock and Rhind, 1991). A GIS is the combination of skilled people, spatial and descriptive data, analytical methods, as well as computer software and hardware (Maguire, 1991; Chou, 1997). With GIS, users have been able to interact with maps in ways that were previously impossible, such as the ability to customise maps through GIS, and the many query and analysis options that are available. This resulted in the creation of a system which was able to automatically manage and deliver geographic information (Zeiler, 1999).

2.1.1 Spatial representations of geographic data

Modelling of real world features requires a reduction in complexity, such that only those aspects of reality the user is interested in remain (Harmon and Anderson, 2003). Even with

the advances in GIS, the options for representing the spatial nature of geographic data remain limited with features most often being represented by simplified geographic representation models, the three most common being rasters, TINs, or vectors (Zeiler, 1999). Rasters were the first spatial data to have been digitally organised (Dangermond and Schutzberg, 1998).

A raster, also known as a grid, is a regularly spaced matrix of cells, with each cell having the potential to hold attribute information as shown in Figure 2.1. The principal purpose of a raster is to represent continuous data such as elevation and rainfall. It can, however, also be used to represent discrete data. A common example of a surface raster model is the digital elevation model (DEM). These models have become a popular and inexpensive way by which to model relief. Unlike vector representations, discrete data are represented as either a single cell (point), a series of adjacent cells (line), or a region of cells (polygon) – all of which have an equal value for any particular feature. As a result of this basic structure, rasters are conceptually simple models. However, datasets are often much larger than those of vectors owing to replication of information and unnecessary detail in some areas, as a result of their recording of values for each cell in their static grid structure (Singh and Fiorentino, 1996; Chou, 1997). The resolution of a raster is important as it is associated with the level of object detail, with resolution being directly related to grid size. This is evident with a high resolution image such as a 0.5 x 0.5 metre DEM displaying more surface detail than a low resolution (100 x 100 metre) DEM, such that small variations in topography are recorded.

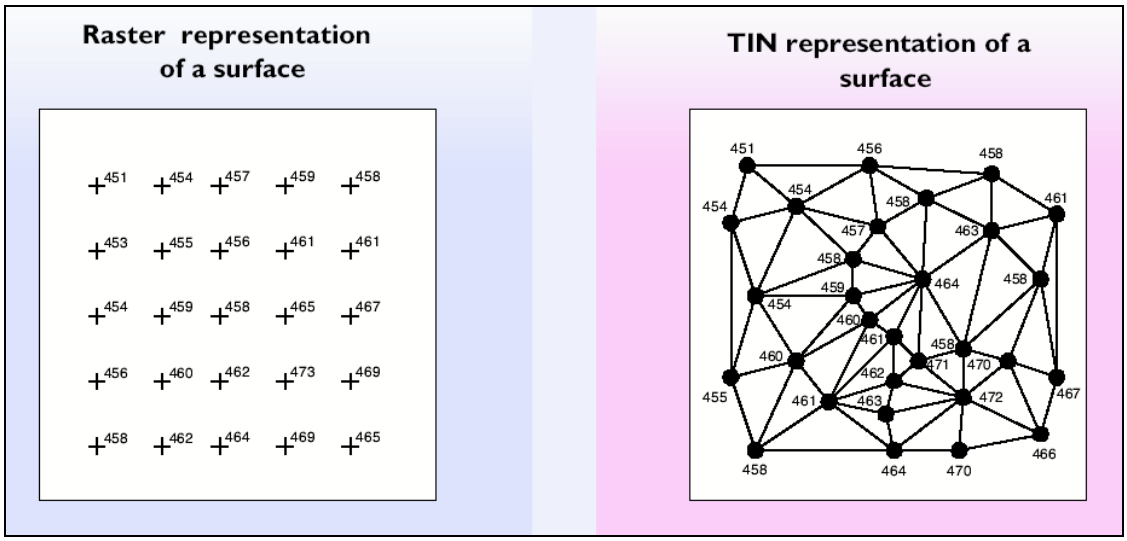


Figure 2.1 Diagram illustrating the difference between raster and TIN representation (Zeiler, 1999)

Although analytically the dominant data model, rasters are limited by their grid structure which, it may be argued, is too rigid to fit to the variable terrain of the real world. This results in features (and in particular linear features), not being well represented, and consequently original captured data are not maintained when interpolating it onto a grid, since it needs to conform to its regular pattern (Zeiler, 1999; DeMers, 2000). Nevertheless, much of the geographic information presently being collected is in the form of rasters captured as spectral data from satellites and other remote sensing sources (Chou, 1997; Rigaux *et al.*, 2002).

When thinking about maps, the concept of a vector representation is familiar to most people. Vector data representations are those maps with discrete features, comprised of the specific geometry types of point, line and polygon. Topographic maps and networks are examples of vector data representations. Each feature is stored as a single coordinate, or as a set of x and y coordinates, with points representing small features, while lines represent features with length and polygons represent areas (Chou, 1997). Features can either be simple, implying that they have no association with other features, or they can be topologically associated. Topology refers to the relationship of connectivity, or adjacency, which a feature has relative to other features. For example, a particular line representing a reach of a river is related to the river network as a whole (Singh and Fiorentino, 1996). Thus for vector data, topologic relationships need to be stored explicitly or must be constructed as needed, otherwise spatial querying, which is fundamental to GIS, is not possible (Harmon and Anderson, 2003). The advantage of vector data are that the geographic locations of features are precise since they can be given exact coordinates (Singh and Fiorentino, 1996). Furthermore, vector databases are relatively small since attributes for a particular feature are only captured once, whilst with raster and TIN databases, attribute values need to be recorded for each cell or point, regardless of whether they share a common feature value (Zeiler, 1999; DeMers, 2000).

As with rasters, TINs are commonly used to model surfaces, and are often used for elevation mapping. Unlike rasters, however, TINs have the advantage of varying the density of recorded surface points, with point density proportional to terrain variability, as shown in Figure 2.1. This reduces data redundancy by only recording those points of importance. Thus in the case of elevation, smooth areas would require few points, while in regions with high topographic variability it would be necessary for more points to be recorded (Chou, 1997; Leung, 1997; Rigaux *et al.*, 2002). A triangulation is then calculated for all the points to create a continuous 3-D surface. A triangulation in this sense is a region of non-

overlapping triangles that completely fill the target area, with each triangle being associated with its neighbouring triangles. TINs are therefore suited to geographic analysis involving elevation, slope, aspect, and volumes. A drawback of TINs is the effort required in data collection, since where topographically variable areas occur, most sampling is necessary, but these areas are simultaneously the most difficult to reach (Zeiler, 1999; Harmon and Anderson, 2003).

Identifying the optimal spatial representation for geographic data is a critical step in any problem-solving task involving the representation of spatial data. The choice as to which representation to use has in the past been presented as an either-or question that debates over the merits of each, and presumes that one representation is better than the others. A comparison between the three main spatial representations described above is presented in Figure 9.1 of the Appendix. For the most part, rasters and vectors have been identified as the premier models of choice. It has, however, now been realised that the needs of a user may incorporate multiple representations, depending on which aspects of the real world are being modelled (Harmon and Anderson, 2003).

As the technology of computers and their associated programs advanced, so too did the art of geographic representation. With the advent of computerised mapping came a change in the way in which geographic information could be displayed. Vector maps were able to be drawn on screens through the use of cathode ray tubes, while rasters could be displayed by using overprinted characters on line printers. Thus the period of the 1960s and 1970s saw the refinement of graphics hardware and software that could produce maps with acceptable cartographic fidelity (Zeiler, 1999).

For over 30 years now maps have been produced with the assistance of geographic data models that enable the digital storage of data that was previously recorded on paper. These data models used one of the three representations mentioned in this section, and were conceived in order to further automate the production of maps, and to enable the storage of a simplified model of the real world to be used for display, query, editing and analysis. As mentioned, the raster model of representation is conceptually very simple, and therefore the associated data models, which include ArcInfo grids, digital images and DEMs, and will not be elaborated upon further in this document. Vector data models are, however, more

complex, with some of the most widely used vector data models being the CAD data model, the coverage data model and the shapefile data model.

2.1.2 The CAD data model

During the 1960s and 1970s when GIS started becoming more popular, computer aided design (CAD) models were the geographic data model of choice. This model was, however, a general purpose model not specifically tailored towards geographic representation, and was a basic graphic without any topology. Since this was the first commercially available model of its type, many data have since been collected in CAD format. It is still commonly used in detailed design and surveying, such as that used in architecture and engineering as it provides a consistent, accurate and digital replacement for technical paper drawings (drafting). The model itself stores geographic data in a binary file format, with specifications for points, lines and areas. The CAD data model is different to its successors in that a single feature may be made up of multiple layers, such that a river could be represented by multiple layers for each river segment (Harmon and Anderson, 2003). Therefore, in a CAD dataset, CAD drawing files are subdivided into CAD feature classes that aggregate all layers for points, lines, polygons and annotation into a single dataset. This layering results in all layers of a similar geometry (such as points) being combined into a common feature class, regardless of their feature type. As a result, differing feature types with a common geometry, such as roads, rivers and boundaries, become assimilated into a single feature class. The CAD data model is most limited, by its inability to store much in the way of attribute data (Zeiler, 1999; Briggs, 2004).

2.1.3 The coverage data model

As GIS became more of a reality, so to did the need for exclusive GIS software. In 1981 the Environmental Systems Research Institute (ESRI) developed the first commercial GIS software; ArcInfo (Briggs, 2004). This software introduced a second-generation data model, *viz.* the coverage, more broadly known as the georelational data model. This has become the dominant model in GIS (Zeiler, 1999), and accounts for the majority of recorded geographic data. The reason for the coverage data model's success was due to its ability to make higher performance GIS possible, by allowing for customisation of attribute tables and through the model's capability of storing topology, therefore assisting in improved geographic analysis

and more accurate data entry (Zeiler, 1999). In retrospect, the model's success could also be attributed to its long-standing prominence as an established and popular ESRI product.

Coverages contain feature classes, which are defined as homogenous collections of features. The primary feature shapes in a coverage are similar to that of vector representations, *viz.* points, arcs (lines) and polygons. The model is distinguished by two characteristics. First, spatial data are combined with attribute data and stored in indexed binary files. Secondly, topological relationships between vector features can be stored, and among other things, provide knowledge as to which nodes delimit a line and thus, by inference, which lines are connected (Zeiler, 1999; Rigaux *et al.*, 2002).

A major advance with the introduction of coverages has been the user's ability to customise attribute tables, thus allowing fields to be added or deleted, and database relationships to be set up to external database tables. As a result, the model's attribute tables can be replaced with a relational database management system (Rigaux *et al.*, 2002). The coverage data model is, however, not without disadvantages. Owing to performance issues as a result of slow processing power of computers and limitations of database software at the time of development, it was impractical to store spatial data directly in a relational database. Thus spatial data were stored in the aforementioned binary files, while related attribute data were stored in the attribute tables, with both being joined through a common identifier. Further limitations include the simplification of features into homogeneous collections of points, lines and polygons. This generic categorisation, for example, results in a road having the same behaviour as a river, which enforces the topological integrity of a dataset by automatically altering features, causing unwanted changes at times (Zeiler, 1999; Briggs, 2004). Thus, if a line were added across a polygon, the polygon is automatically split, which would be undesirable if no change to the original polygon were intended. A feat to be achieved with future data models was the ability to support special behaviours of real world features such as river and roads. This is illustrated by the ability to have the volume of flow of two rivers joining to become the addition of both. Although there had been some success with the coverage model in this respect, it became apparent that a better way was needed to associate behaviours with features (Zeiler, 1999).

2.1.4 The shapefile

While topological datasets, such at those found in coverages, provide for complex geographic analysis and map display, many uses can be satisfied with a simpler form of feature data (Zeiler, 1999). Recognising the need for an additional data model that supported simple feature classes, ESRI introduced the geographic data model known as the shapefile (Zeiler, 1999; Briggs, 2004). In this data model, simple feature classes store the shapes of features, but do not store their topological associations, and have the advantage of simplicity and rapid display performance. However, by not being able to enforce spatial constraints through topologies, validation functions such as wanting to ensure no gaps exist between polygons, are not possible. Other than this, shapefiles are similar to coverages in many respects and thus were not thought of as a revolution in the modelling industry, although the shapefile has since become widely used. This data model provides the framework for a simple feature dataset and allows for map display and query.

Shapefiles are composed of multiple files that contain spatial and attribute data. These are the *.shp* file which stores feature geometry, the *.shx* file which stores coordinate geometry and the *.dbf* file which is a dBASE file which stores attribute information (HDS, 2001). An additional dBASE table is sometimes used for the storage of attribute data relating to objects that do not necessarily have a spatial context, yet are related to a particular spatial feature. This is illustrated in the case of owners of a parcel of land, where the parcel of land has a geographic reference, while the owner has a spatial context by association. Like the coverage, a shapefile is a vector representation that uses a homogenous collection of point, multipoint, polyline and polygon layers (Zeiler, 1999; DeMers, 2000). A comparison between the three ESRI data models capable of storing vector data is presented in Figure 2.2. These data models represent a progression of functionality and detail in the storage of geographic data, leading up to the development of the geodatabase, which is included for the purpose of comparison although the geodatabase is only discussed in detail in Section 2.3. Figure 9.2 in the Appendix, provides additional information, as to the differences between the geographic data models in the way they store feature topology and geometry.

	Coverage	Shapefile	Geodatabase
Data Collection	An ArcInfo workspace is a collection of coverages, grids and TINs.	A folder can contain shapefiles.	A geodatabase is a collection of feature datasets, rasters and TINs.
	Spatial data are stored in binary files. Topological and attribute data are stored in INFO tables.	Spatial data are stored in binary files. Attribute data are stored in dBASE tables. No topological data are stored.	All spatial, topological, and attribute data are stored in tables in a relational database.
	For large datasets, coverages are subdivided into tiles in a map library.	Shapefiles are continuous for small to moderately sized datasets.	Geodatabases span continuous geographic extents.
	Behaviour is loosely coupled with features	Behaviour is loosely coupled with features through VBA macros.	Behaviour is tightly coupled with features through rules and code written for custom feature classes.
Feature Dataset	A coverage contains topological feature classes that participate in line or polygon topology.	A shapefile has one simple feature class.	A feature dataset in a geodatabase contains simple or topological feature classes.
	Line topology is implemented with arcs, nodes and routes. Polygon topology is implemented with arcs, label points, polygons and regions.	Polygon topology among a set of shapefiles can be implemented with on-the-fly topological editing.	Line topology is implemented through a geometric network. Polygon topology is implemented through on-the- fly topological editing.
	Only one feature class is associated with a topological role.	There is no implicit topological role for a shapefile.	Many feature classes can be associated with a topological role.
	No associations are defined except for topological associations among related features like arcs and polygons.	No associations are established among features in shapefiles.	User-defined associations can be established between features indifferent feature classes.
	Coverages have a defined coordinate system.	Shapefiles have no defined coordinate system.	Feature datasets have a defined coordinate system.
Feature Class	A coverage feature class stores feature geometry in a binary file and attributes and topology in a feature attribute table.	A shapefile stores feature geometries in a binary file and attributes in a dBASE file.	A feature class stores features in a relational table with a special field for the geometric shape of a feature.
	The primary coverage feature classes are point, arc, polygon, and node	The types of shapefiles are multipoint, point, line and polygon.	The types of feature classes are point, line, polygon, annotation, simple junction, complex junction, simple edge and complex edge.
	A coverage feature class cannot be extended.	A shapefile cannot be extended.	A feature class can be extended to a custom feature class.

Figure 2.2 A comparison of the three major geographic data models (Zeiler, 1999)

2.1.5 Limitations of the preceding geographic data models

The review of the CAD, coverage and shapefile in the preceding sections provides some insight into how geographic data models have changed over the years. The options for spatial representation of geographic data have, at times, created a division in spatial modelling, such that multiple geographic representation models could not be used concurrently. In addition, as the previously reviewed coverage and shapefile data models demonstrate, geographic data models in the past have used a combination of spatial and attribute data that have been stored in separate file formats. In recent years, ESRI has developed and introduced a new geographic data model known as the geodatabase to overcome these aforementioned limitations. Two of the other main benefits of the geodatabase are the storage of both spatial and attribute data in

a single database (Harmon and Anderson, 2003), and the tight coupling of behaviour with features (Zeiler, 1999).

2.2 Database Models

In addition to the advances made in geographic data models, there were concurrent advances being made in the field of database models, which will be outlined before detailing the geodatabase. A database model may be described as a collection of logical constructs used to represent the data structure and relationships found within the database (Rob and Coronel, 1997). A data structure is defined as the logical arrangement of data as used by a system for data management and is the schematic representation of how the data will be stored for retrieval (AGI, 2006). Early GISs all used the layer as the principal data structure, which came from the past use of layering to produce analogue maps. This method is a longstanding practice in GIS and has some advantages including the ease in which suitable layers can be combined to produce maps, and how any unnecessary layers can be turned off during map query. The issue at hand, however, is not that of display, but rather of representation in the database, with two particular innovations paving the way for the development of the geodatabase, *viz.* the object-oriented and relational database models (Harmon and Anderson, 2003).

Databases are an important component of any GIS and are ordinarily classified according to the data structure they use. Thus, the data structure of a geographic data model has traditionally been mapped to the user's view of the data through the use of network, hierarchical or relational database models. Both the network model which permits the modelling of many-to-many relationships in data, and the hierarchical model which organises data into a tree structure have since become outdated, while the relational model is still widely used. The emergence of object-oriented modelling has allowed for a new way in which to map a conceptual schema to its logical schema, and utilises the concepts of the network and hierarchical models through the use of its relationships and tree data structure respectively (Raza, 2001).

2.2.1 The relational database model

The development of the relational database model came about through the concepts first set out by E.F. Codd in 1970 concerning the relational model (Healey, 1991; Rob and Coronel, 1997). In comparison to previous approaches, the relational database model can perform the same basic functions as the hierarchical and network database models. The relational database model benefits, however, from an independent structure not found in the hierarchical and network database models, such that the working of the database model is not affected by any changes in the database schema (Rob and Coronel, 1997). Yet this database model is characterised by simplicity in that it is merely perceived as a collection of related tables, while each of the tables is a matrix consisting of a finite series of rows and columns (Healey, 1991; Rob and Coronel, 1997; Raza, 2001). The characteristics of the relational table have been elaborated upon by Healey (1991), Worboys (1995); cited by Raza (2001), and Rob and Coronel (1997) and are as follows:

- Each row in a table represents the data for an individual entity;
- Each column in a table contains data for any one attribute and has a distinct name;
- The ordering of rows is not significant; however, each row must be distinct from the others;
- All values in a column must conform to the same generic relational data type such as integers or text; and
- Values for an entity are explicitly stored at the intersections of rows and columns.

A relational database consists of a set of tables, each with its own relational schema. The principal features of relational databases are the primary key and relational joins. Since each row must be distinct from the others in a relational schema, it follows that a single column, or a combination of columns, can be used to define a primary key. The primary key allows each row to be uniquely identified and can thereby be used as an addressing system for a particular table. No physical links exist between tables in a relational database and therefore the only way to join or relate one table to another is through a relational join. This involves matching values in a column of one table to values in a column of another table. Matching is frequently based on the primary key of one table being linked to a column of another table, which is termed the foreign key. Cardinality restricts the number of relationships that can be formed between the primary key and the foreign key. Cardinality therefore refers to the type of

relationship found between columns in database tables or between features in a geographic dataset. This cardinality of a relationship may be one to one, one to many, many to one and many to many (Zeiler, 1999), as seen in the Figure 2.3. When working with a relational database, relational joins can be used over multiple tables until the necessary data from the requisite tables have been retrieved (Healey, 1991; Rob and Coronel, 1997; DeMers, 2000; Raza, 2001).

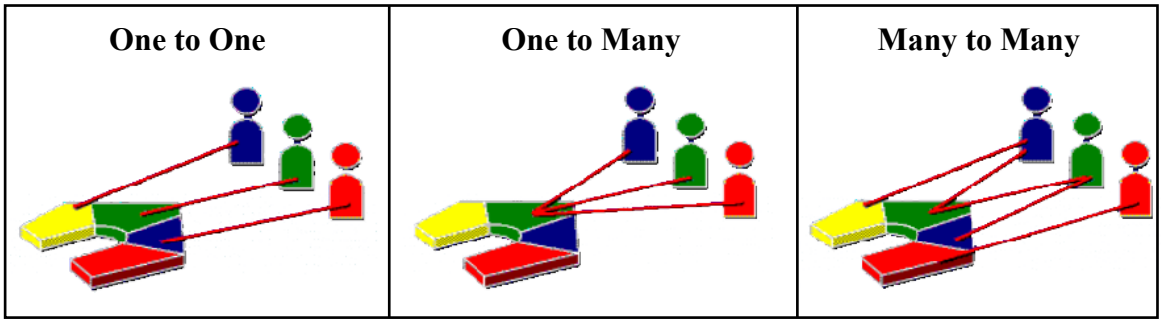


Figure 2.3 An illustration of the basics of cardinality (ESRI, 2002)

As a result of the inherent relational algebra, a major advantage of the relational model is that it enables queries to be performed on a database. This algebra consists of a set of operations, designed to work with the generic relational data types, and is used on tables to produce new relations (O'Neil, 1994; cited by Raza, 2001; ESRI, 2004). These query capabilities and operations are based on the Structured Query Language or SQL (Raza, 2001).

As database management software has matured, so to have the advantages of its combination with GIS become more established. This trend has been well marked within the relational database management systems (RDBMSs) that have traditionally dealt with more tabular data types, such as attribute data, rather than spatial data, as will be detailed in subsequent chapters. This integration of GIS with relational databases demanded the development of new middleware technology, of which the geodatabase is an example. The combination of this new technology with enhanced RDBMSs allows for the storage of the different spatial data types, and provides access and retrieval tools. Dangermond and Schutzberg (1998) point out two distinct options when considering the union of spatial data with a relational database:

- Storage of spatial data in the RDBMS can be hidden, such that the software cannot distinguish it from other data (such as Binary Large Objects). Middleware software is then used to index, query and perform spatial analysis of the spatial data, as is the case with the geodatabase; and

- A new spatial data type is added to the database via a plug-in technology. This new data type is fully registered in the database, along with its own set of spatial functions. By July 1998, when this approach was quite new, it was only supported by a small number of the commercial RDBMS's; *viz.* Informix's Dynamic Server and IBM's DB2 Universal Database.

Whether spatial or other, the storage of data in a relational database provides a number of benefits, including the previously mentioned querying capacity. Other benefits include its ease of use and implementation, reduced data redundancy from the elimination of much duplicate information and simpler modification compared to other database models (Healey, 1991). Added to this are the management aspects of a RDBMS, which include security, backup, client/server control, and the benefit of working in an application development environment (Dangermond and Schutzberg, 1998). Additional information about the features of a RDBMS is provided in Table 2.1. By storing large amounts of both spatial and non-spatial data in a single RDBMS, data sharing is facilitated and can more readily result in the establishment of large GIS projects (ESRI, 2003a).

2.2.2 The object-oriented database model

Since the 1990s, object-oriented modelling of geographic data has been gaining attention, and has been used in the development of object-oriented databases. This extension of object-orientation to databases was a logical outgrowth of the object-oriented programming style that started gaining popularity during the 1990s (Harmon and Anderson, 2003). The fundamental idea behind the object-oriented approach is to allow the user to model data in terms of real world objects that interact with each other, rather than as computer records or tables (Batty, 1991).

Rob and Coronel (1997) define object-orientation as a set of design and development principles based on conceptually independent computer structures known as objects, with each object representing a real world entity with the ability to interact with itself and with other objects. An object is an instance or occurrence of a class, where a class is defined as a description or specification of a group of objects with similar attributes, common behaviour, and relationships with other objects (Rob and Coronel, 1997; Twumasi, 2002). In an object-

oriented system, an object is defined by three primary traits (Rob and Coronel, 1997; Twumasi, 2002), *viz.*

- Its *identity,* which uniquely distinguishes it from other objects;
- Its *state*, described by the values of its attributes (Mitášová and Višòovcová, 2002); and
- Its *behaviour,* which describes operations performed by the object and the way it interacts with other objects (Mitášová and Višòovcová, 2002).

The object-oriented database model is characterised by three key qualities. The first is *polymorphism*, which deals with the ability of an object class to adapt its behaviour in relation to other objects. The second is *encapsulation*, which means that an object is only accessed through a set of well-defined software protocols, thereby enforcing the use of a database management system. The third trait is that of *inheritance*. This is of particular importance since it enables an object class to include the behaviour of parent classes along with its individual characteristics in a top down approach (Wachowicz, 1999). This results in reduced data redundancy and more effective data modelling in terms of speed and simplicity (Zeiler, 1999; Rob and Coronel, 1997; Rigaux *et al.*, 2002).

Object-oriented modelling and design are based on the concept of objects, their interactions, relationships, and organisation. Similarly, these concepts form the basis for object-oriented database management system (OODBMS), which integrates object-orientation and database functionality (Khoshafian, 1993; cited by Twumasi, 2002; Rob and Coronel, 1997). Object-oriented databases provide a new way in which to conceptualise and implement the storage of data, and to overcome some of the limitations of the relational data structure. Table 2.1 provides a comparison between object-oriented and relational databases. With regard to the application of database models to the spatial domain, the OODBMS allows for a map to be more than just a visual interface, as it provides an understanding as to the real objects behind the assembly of graphic elements (Mitášová and Višòovcová, 2002).

Table 2.1 A comparison of relational and object-oriented databases (after Raza, 2001)

	Relational database	Object-oriented database
Navigation	**Slow**: Joins are inefficient for extensive navigation.	**Fast**: Navigational queries are as fast as accessing data in memory because of smart caching strategies.
Advanced features	Lacks advanced features: ***Inheritance***: No inheritance mechanism. ***Schema evolution***: Quality of database to change the structure of database populated with data ***Versioning***: Various states of an object. ***Configuration***: Set of mutually consistent objects. ***Long transaction***: A series of database commands that extend over a long period of time (days, weeks, months etc). RDBMS provides short transactions normally executed within few seconds. ***Change notification***: Ability of database to notify the user whenever an object is changed.	
Data types	**Few**: New data types cannot be defined.	**No limit**: Many (complex) data types can be constructed from primitive data types, such as point from x and y data types.
Paradigm	Table: Sometimes data do not naturally fit within the confines of a table and decomposition is required.	An object naturally fits in a file / table. No decomposition is required.
Theory and standard	Based on formal theory of relation and SQL standards. Therefore, products are standardised. Vendor independent.	Not based on formal theory, therefore products are not standardised and vary from vendor to vendor.
Availability	Owing to standard and well-known data structure, relational model is widely deployed.	Not widely available.
Extensibility	**Much extensibility**: The ability to change database schema without changing the applications programs.	**Lack of logical extensibility**: Existing products lack logical data independence; not a flaw of OO databases but rather of commercial products.
Database corruption	Database and applications run in separate processes; therefore chances of data corruption are less, although some performance is lost.	Database is run in the application process space. Therefore, database runs the risk of security violation or corruption by wild pointers.

2.2.3 The object-relational database model

Both the object-oriented and relational approaches have their strengths and weaknesses, as shown in Table 2.1. The combination between these two modelling approaches has been researched by many commercial database vendors and has led to the creation of a new database model known as the object-relational database model. This model, also called the extended relational model, is a compromise between the concepts set forth in both models and aims to incorporate object-oriented features into a relational database. Thus many of the shortcomings of the relational database can be overcome, whilst still including the simplicity and functionality of the relational database model (Raza, 2001; Twumasi, 2002).

The object-relational database has become increasingly popular for the storage and query of geographic data, gradually replacing the traditional layered databases, as illustrated by the decline in use of the CAD and coverage data model in GIS. Many software companies have adopted object-relational technology for the storage of geographic data, and each has its own terminology. Harmon and Anderson (2003), refer to this type of a database as a (SERD). A SERD inherits many of the characteristics particular to object-oriented and relational modelling, and applies them to spatial and attribute data. Modelling of geographic features in a SERD requires that the database adhere to the following general principles (Harmon and Anderson, 2003):

- Hierarchical organisation: Objects belong to a hierarchy with their position being of importance, since objects lower in the hierarchy belong to and are influenced by the objects above them;
- Objects have properties: A feature must have a property type, such as number or text depending on what information is being recorded;
- A spatial object in a geographic database has location: In a layered geographic database it is common for a feature to have two types of information, namely spatial information and attribute information. In an object-oriented database this information is stored collectively in a common object, and thus the idea is that location is just another property of an object. This means that when an object is deleted in a database, so too is its attribute information;
- Objects properties can be affected by their spatial context – An object may have different properties relative to its existence in space. For example, by a change in scale, a river in a catchment may appear as a simple line, but when magnifying a particular section, braids and channels may become apparent;
- Relationships between objects have properties: Objects can relate to one another and their relationships can have properties governing the way in which a change in one object impacts a related object. This adds behaviour to a geographic database where, for example, a control valve for a water network regulates the flow to the receiving end of the network it is part of;
- Objects belong to classes: There are an infinite number of objects in reality, that when viewed individually, are impossible to deal with in a geographic database. Logical groups of objects are, however, normally distinguishable and would fall into

appropriate classes. For example, the class called river could be split into sub-classes of perennial, intermittent and ephemeral, each with their own properties; and

- Properties can be inherited: Combining the idea of hierarchical structure and properties, object-oriented systems are designed so that subclasses inherit the properties of the class to which they belong. Therefore, the subclasses of the river class might inherit the properties of river length, width and depth. Nonetheless, all could have unique values for each of these properties.

In the same way that object-oriented programming languages have become the new standard, so too object-oriented geographic databases are likely to eventually replace layer-based databases (Harmon and Anderson, 2003).

2.3 The Geodatabase Data Model

With the increasing utilisation of GIS applications in various industries, and the realisation that a better way was needed to associate behaviours with features, came the requirement for a new geographic data model. The coverage model was only partly successful with adding behaviour to features through external code, and so further development of the model was considered. This, however, was not viable as the new requisites for the application code and feature classes were almost impossible to deal with jointly, as synchronicity between the two had to continually be updated. Thus a new geographic data model with an infrastructure to tightly couple behaviour with features was developed, known as the geodatabase (Zeiler, 1999). Past methods of storing spatial and attribute components of geographic data in separate files were undesirable owing to the disaggregation of related data. Accordingly, the new geographic data model places both spatial and attribute data in the same database, and combines the benefits of the relational approach with those of object-orientation, to create an object-relational model – ESRI's interpretation of a SERD (Harmon and Anderson, 2003).

Zeiler (1999) describes the geodatabase model as the bridge between peoples' cognitive perception of the objects surrounding them in the world and how those objects are stored in relational databases. The definitive purpose of the geodatabase data model is to enable the features in a GIS dataset to attain greater functionality through closer association with real world objects, by providing them with natural behaviours, and by allowing any sort of relationship to be defined between features. Thus the geodatabase brings a physical data

model closer to its logical data model. The geodatabase also has the ability to implement the majority of custom behaviours without any further programming. This is made possible by the various ways in which behaviours can be defined through options, including domains and validation rules (Zeiler, 1999). The following section on the object-oriented design of the geodatabase further explains many of these concepts

The introduction of ESRI's SERD was not a concept exclusive to the company. Both the software companies Intergraph and Oracle introduced their own interpretations of a SERD around the same time. Owing to reasons including comparatively little marketing and the high cost of products, these solutions have not been as successful as the geodatabase (Harmon and Anderson, 2003).

2.3.1 Object-oriented design of a geodatabase

The geodatabase serves as a generic model for geographic information, as well as providing a framework for the creation of an object-relational database schema and the implementation of behaviours for objects in the database (Zeiler, 2001). Object-oriented concepts of inheritance, encapsulation and polymorphism are supported by the geodatabase. Thus the object-relational data model upon which a geodatabase is built, results in a number of important characteristics (Zeiler, 1999):

- Adding and editing of features can be controlled through the use of built in functions such as domains, that permit certain values to be assigned to an attribute, and connectivity rules, which determine whether one feature can be placed adjacent or connected to another feature; and
- Relationships among features can be set up including topological relationships in a connected system such as a river network, or general relationships such as a parcel of land and its relationship to its owner.

The use of object-relational database modelling provides advantages such as creating and editing features and feature relationships, as discussed previously. There are, however, further benefits that are evident when using the geodatabase data model and these include the following (Zeiler, 1999; Evans and Millett, 2002):

- All geographic data (including raster and vector) are held in a uniform repository and can thus be managed as a single database;
- Data entry and editing is faster and more accurate through the use of data rules and relationships and other validation methods;
- Since features and objects are assigned behaviour, a geodatabase can now correspond to the user's model of data instead of generic points, lines and polygons, thus making the modelling process more intuitive, through the closer association of objects with reality;
- The behaviour of features is further added to by polymorphism. This means that features on a map display are dynamic in that they respond to changes in neighbouring features;
- The relational database upon which the geodatabase is built, allows for the simultaneous editing of geographic data by multiple users, enabling the management of large databases (versioning);
- Complex feature interactions can be modelled through the creation and editing of geometric networks and topologically integrated features; and
- The data model enables the development of industry-specific models such as the Arc Hydro data model.

To state that these advantages are available solely to the geodatabase model would be incorrect. Many of the aforementioned characteristics could have been achieved through previous data models, but this would only have been possible through the creation of external code and therefore added complexity. An additional advantage of the object-oriented modelling nature of the geodatabase is that the object view of data allows one to focus on building the data model, while hiding much of the underlying physical structure of the data model (Zeiler, 1999).

A geodatabase does have potential constraints, created by the added complexity of the new data model to its forebears. An example of this is where the single repository of geographic data in the geodatabase, and the data validation built into the model, sometimes make the exportation of subsets of geographic data more difficult. The use of a geodatabase can also result in added complication, especially when the user does not require the advanced functionality offered by the new data model.

2.3.2 How a geodatabase extends a relational database

The geodatabase is built upon the relational database model and is comprised of tables containing spatial and attribute data. A geodatabase's logical elements are represented by a relational model's database elements, as shown in Table 2.2. This provides some insight into how the geodatabase data model uses tables and rows as a substitute for classes and objects, in order to implement the object-oriented method. The main improvement that results from the interaction between a GIS and a RDBMS is that their combination allows for efficient storage of geographic data (Batty, 1991). The common functions of a relational database management system are therefore increased when incorporating a GIS. These include the ability to store the geometric shape of features, the inclusion of a spatial index for rapid retrieval of geographic information, and the use of versioning (Zeiler, 1999). Thus, as the name implies, a geodatabase can be seen as the definitive instance of a GIS and a relational database.

Table 2.2 Logical elements of a geodatabase with associated relational database elements (Zeiler, 1999)

Logical Elements	Database Elements
Class	**Table**
Attribute	**Column, Field**
Object	**Row**

An important factor to consider when applying a geodatabase is the RDBMS it uses, as this affects the geodatabase's functionality. In this regard, two instances of a geodatabase exist, *viz.* personal and multi-user. Figure 9.3 in the Appendix provides a summary of the differences between the two. The ESRI product being used is an important consideration, as only the multi-user Arc Spatial Database Engine (ArcSDE) can be employed to create and manage both types, as it provides the necessary gateway between the GIS and the DBMS to share and manage spatial data.

When dealing with the geodatabase it is possible to disaggregate it into a number of components. The collective and primary unit of geographic data is the geodatabase itself. It is, however, not necessarily the case that all geographic data for a particular project are

present within a single geodatabase. Data are often sorted into multiple geodatabases according to broad categories such as transportation or environment, with the geodatabase being structured in such a way as to maintain the geographic data as a whole unit, so that there is no partitioning of data.

The next level of division is that of the geographic dataset within a geodatabase. This dataset is divided according to the geographic representation models of vector, raster and TIN. The vector based feature dataset will be used as the example dataset for the remainder of this section, as it is the most complex yet customisable of the three, and thus demonstrates the geodatabases structure best (Zeiler, 1999). A hierarchical view of a geodatabases structure is shown in Figure 2.4, and the various components making up a geodatabase are shown in Figure 2.5.

The subsequent division of geographic data is that of classes. Classes are divided into object classes, feature classes and relationship classes, and they can exist either within or external to a feature dataset. Object classes are simply data tables within the geodatabase that store non-spatial descriptive information about entities, yet are related to geographic features. The owners of different parcels of land would constitute an object class since, although their geographic location is not recorded, a relationship exists between them and the spatial data as represented by the parcel of land they own. As with features, objects of the same kind are grouped within an object class. A feature class is thus a collection of similar features with the same type of geometry. Thus in terms of feature datasets, feature classes are split into distinguishing groups of point features, line features and polygon features. A feature class can additionally be split into two categories. The first is a simple feature class where no topological associations exist between features. The second is a topological feature class, which serves to bind together features that form an integrated topological unit, and is illustrated by individual river reaches being bound together to form a network. Finally, relationship classes are present in a geodatabase as tables that store relationships between features and objects (Zeiler, 1999).

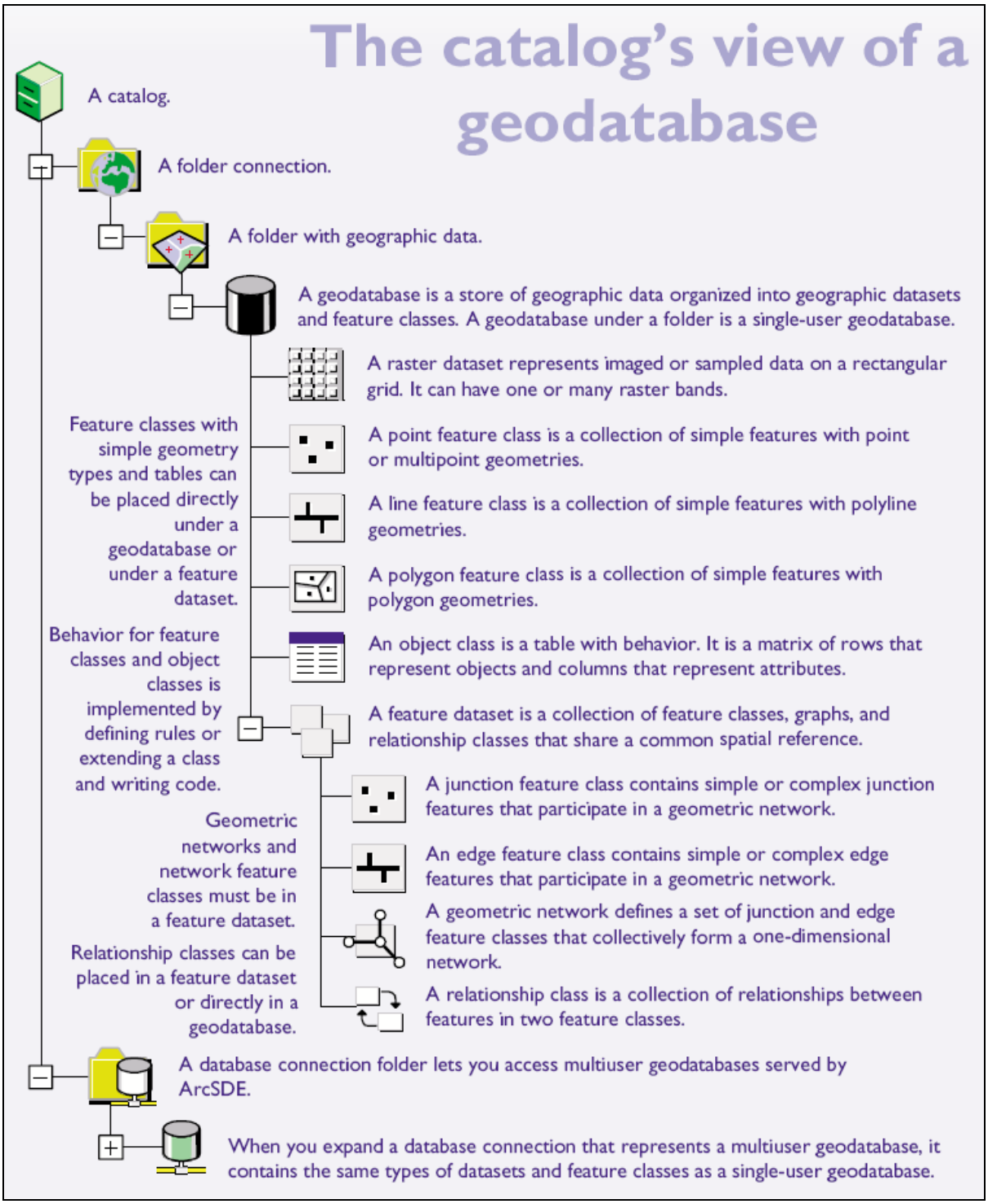


Figure 2.4 A hierarchical view of the geodatabase, displayed using ArcCatalog (Zeiler, 1999)

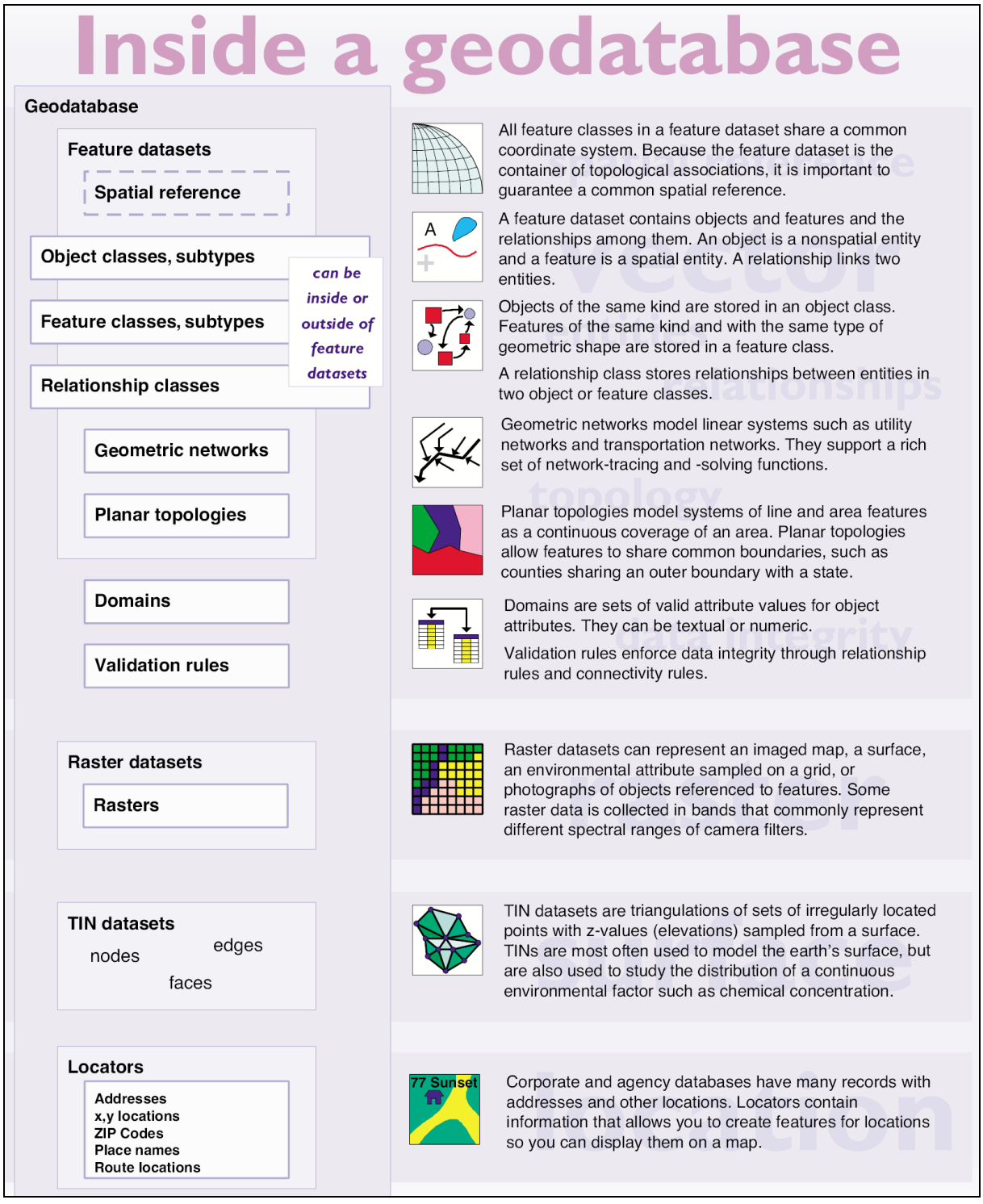


Figure 2.5 The components making up a geodatabase (Zeiler, 1999)

The geodatabase is an example of a unified data model in that it is created with the ability to incorporate data in many formats and from different sources. Previous geographic data models and the data they include, such as coverages and shapefiles, can either be incorporated within, or be upgraded, to become part of a geodatabase. Thus a geodatabase provides the ability to have a uniform set of tools with which previous geographic data models can be

managed. In this respect, a geodatabase can contain four representations of geographic data. These are vector data for representing features, raster data for representing grids, TINS for representing surfaces, and addresses and locators for finding geographic position (Zeiler, 1999). Details particular to the first three of these in relation to a geodatabase will be explained in the following section. Addresses and locators will be excluded, since they pertain to finding a geographic position, and are not required for the remainder of this study.

2.3.3 Rasters and TINs in a geodatabase

Rasters, which were reviewed in Section 2.1.1, are a common representation of geographic data and can be integrated into a geodatabase. Each cell attribute of a raster defines the class, group, category or measure at the cell position, with cell values being recorded as either integers or floating points. Cells with identical values can be placed in the same group, and thus by using a one to many relationship, a single grid cell value can correspond to a group and, by inference, to a number of individual cells. This can result in a reduced amount of storage space used by a raster. At high resolutions, raster datasets can become hardware intensive owing to the large amount of disk space used as a result of the added detail. Thus raster pyramids are created in order to down-sample a raster to a sufficient resolution for a particular scale. These pyramids therefore create instances of the same raster by averaging cell values at different magnitudes of scale, with correspondence to the level of detail required. Thus, when viewing a particular raster at a large scale, a higher resolution would be required as opposed to when viewing the same raster at a smaller scale (Zeiler, 1999).

TINs are another geographic data representation that can be incorporated into a geodatabase. Both raster and TINs are used in surface modelling. However, TINs are often more beneficial in this respect because of their ability to represent a variable surface more accurately. Unlike the uniform grid structure of a raster, TINs require a slightly more complicated method when trying to model surface feature. For point surface features, a TIN uses preserved nodes from triangulation, while for line surface features, connecting lines between nodes are used. Uniform areas are defined by replacing individual triangle values, with a constant z value. This produces a particular polygon according to the triangles amalgamated (Zeiler, 1999). Refer to Figure 2.1 for clarity on the TINs structure.

2.3.4 Features in a geodatabase

Objects in the real world have a systematic set of rules and relationships that they must follow. For example, a river must flow downhill and plants must transpire in order to grow. The geodatabase tries to incorporate these rules and has been most successful in the modelling of vector features. Features in a geodatabase have many qualities such as shapes, relationships, attributes and behaviours. All of these create a rich context by which a feature can relate to its location or neighbouring features and the response it would have from a change in these. The shape of a feature is stored in a relational table field called *shape*, as either a point, multi-point, line or polygon. Each feature may have a number of properties defined, such as its spatial reference, attributes, subtypes and relationships (Zeiler, 1999).

When developing features in a geodatabase, it is first necessary to account for all the types of objects that are being modelled and the feature classes that represent them. This allows for the creation of feature datasets that groups feature classes together. Some reasons for the creation of a feature dataset include those instances when feature classes share a common spatial reference, when they have topological associations as in a network, or when they have similar thematic content such as land use. As mentioned previously in Section 2.3.2, there are three basic types of feature classes, these being object, feature and relationship classes. The feature classes in a feature dataset all share a common spatial reference. Features have a geometry made up of x and y coordinates, with optional z (for height) and m (for distance) values (ESRI, 2002; Evans and Millett, 2002). These coordinates relate to a spatial reference inferred from the shape of the earth (Zeiler, 1999).

Attributes characterise the qualities of an object, or feature, with descriptors such as name, area and location. Features on a map can have any number of associated attributes. Apart from providing the characteristics of a feature, attributes can also define subtypes of a feature. A subtype is a special attribute that allows one to assign distinct simple behaviour to different classifications of objects or features (ESRI, 2002). Such a classification might be used for a stream system, with subtypes of 1st, 2nd and 3rd orders. All subtypes share the same attributes, but specific behaviour is possible through the use of distinct attribute domains, default values and relationships. It is an important design decision as to whether a group of related features should become a set of feature classes or a set of subtypes of a feature class.

A key reason for the use of subtypes is their ability to improve performance and to help enforce data integrity through domains and validation rules (Zeiler, 1999).

The basis for the use of domains and validation rules is to eliminate or minimise error in data entry. An attribute domain is a specified set or range of valid attribute values that prevent simple mistakes in data entry and editing. Range domains allow for numerical values between a maximum and minimum value, while coded value domains are used to define a particular set of values which can then be recorded as either numbers or text (ESRI, 2002; Evans and Millett, 2002). A third option in domains is to have a default value, thereby enforcing some specified value when none is available. Integrity of datasets is further ensured by the use of validation rules that control features and attributes. Three types of validation rules are described in Zeiler, 1999:

- *Connectivity rules* decide whether one feature or object can be connected to another, such as a 20 mm valve only being able to be connected to a 20 mm pipe;
- An *attribute rule* is an attribute domain applied to a subtype of a class. Thus, hypothetically speaking, a vegetation class may have a subtype of sugarcane, with an evaporation attribute domain of between 0 and 10 mm a day. Attribute domains can have additional rules attached. These are present as rules for either splitting or merging features; and
- *Relationship rules* constrain cardinality (illustrated in Figure 2.3) of a relationship between an origin class and a destination class.

Objects interact with each other in several ways. The association between objects can be maintained by topological links or through relationships. In the geodatabase, relationships are organised into relationship classes where each relationship in a set class has the same origin and destination class (Evans and Millett, 2002). It is thus desirable to keep track of relationships so that when one object is modified, the related object can react appropriately (Zeiler, 1999). The object-relational model upon which the geodatabase is built provides the facility for this to be done.

2.3.5 Geometric networks

Networks are an essential part of our reality, as seen in the case of network infrastructure such as roads, energy, pipelines and commodities. The structure, capacity and efficiency of networks have a significant influence on our lives. The introduction of ESRI's geodatabase also debuted the geometric network. This new model was developed by ESRI from extended experience with transportation and utility networks. The primary advantages of this new model are that it makes editing easy through the use of connectivity rules, and allows network features to represent complex features such as switches, thereby reducing potential clutter (Zeiler, 1999).

Networks are made up by two types of objects, *viz.* edges and junctions. These are equivalent to the arcs and nodes in a coverage. A geodatabase has a dual network representation system in place. These are the geometric and logical network. As with standard vector line features in a geodatabase, geometric networks share the same spatial and attribute characteristics, but with the addition that they can preserve connectivity between lines and that they automatically update network elements. Geometric networks are, in turn, connected to logical networks, which are a collection of connected edges and junctions with the purpose of storing connectivity information and some attribute information. The key difference between the two types of networks is that in the case of a logical network there is no corresponding spatial reference, while the geometric network stores the location data explicitly (Zeiler, 1999).

Connectivity rules play an important part in networks, with established rules relating to connections between edges, junctions and edge-junctions. Consequently, cardinality is a factor. Networks can be split into two key operational categories, *viz.* utility and transportation networks. In a transportation network, the entities using the network have choice. Thus in the case of a road, a driver might turn left or right. In a utility network, the entity's direction of flow follows the rules of the network, such as water in a river flowing downstream. The changeable direction of any movement is controlled through the use of switches, sources and sinks, with the direction of movement in edges being established through interpretation of the layout of these controllers. The reason for this distinction is that a switch dictates direction of movement, while a source (such as the spring of a river) is the start of movement and a sink (such as a river estuary) is the end (Evans and Millett, 2002). The ability to disable or enable features further adds to the control one can have over a

network, since this allows certain routes to be opened or closed. Weight is also an important factor to consider when discussing networks. Weight is recorded in a logical network and is used to store the cost of moving through a network. It is commonly presented as distance; however, any numeric field can be a weight. Sometimes a decision needs to be made which is based on a network issue, such as finding the shortest route. Network analysis is thus a useful way to solve such problems or queries. This is achieved through queries that navigate the network and produce a result, such as finding shortest path or all elements upstream of a point (Zeiler, 1999).

2.3.6 Geodatabase design

A GIS has the potential to achieve a number of tasks at a range of spatial and temporal scales, from daily operations at a local scale, to long-term planning at a national scale. This is made possible by simple operations such as storage and dissemination of data, to complex integration with other technologies, achieved through the open architecture of object-oriented models. What makes the implementation of such a GIS most effective is good design. Traditional relational database design is characterised by the conception of a logical data model and subsequent implementation of a physical database model. Thus a logical data model captures the user's view of the data, whilst the physical database model implements the data model within the structure of a RDBMS (Zeiler, 1999).

The objectives of the design process are to define goals, identify and evaluate alternatives and create an implementation plan. A design allows one to know the stage of development and how to progress, while keeping an overall holistic view. As the design progresses, so to does the amount of detail, such as adding data definitions or assigning spatial structures. The characteristics of an efficient geodatabase include meeting organisational requirements and objectives, reducing redundancy of data, creating alternate views of data and organising geographic features appropriately. The creation of a geodatabase design can, however, be quite time consuming and benefits from the following (Zeiler, 1999):

- Involving users, as this will create a sense of ownership and identify user requirements;
- Using a multidisciplinary team of experts to achieve specific design objectives;
- Setting up milestones that can be used to gauge your progress;

- Keeping a holistic view of the project at all times;
- Only adding detail once the basic structure of the geodatabase is in place; and
- Planning from your model in order to meet your organisation's priorities.

Zeiler (1999) divides conventional geodatabase design into five steps. The first three steps develop the conceptual model by classifying features and their spatial representations, while the fourth and fifth step develop the logical data model and integrate the conceptual model with the geographic datasets. The construction of a logical data model is a repetitious process, with no single perfectly correct model, but rather better and worse ones. Determining whether a data model is good enough is a question best answered by looking at whether it has met the demands of the user with as little data duplication as possible. Upon completion of the logical data model design, the next step is to implement it in a physical database model. A benefit of the geodatabase in this respect is that it allows the structure of the physical data to closely match the logical data model (Zeiler, 1999). Arctur and Zeiler (2004) include a similar stepped design procedure, but expand the five-step design by Zeiler (1999) into a ten-step design, as shown in Table 2.3.

2.4 A Geodatabase Application: The Arc Hydro Data Model

Many forms of geographic information are required for the management of water resources, including catchment delineations, rainfall data and river connectivity. The purpose of this section is to outline the use of geodatabases in a hydrological context, as well as to provide an example of an industry-specific geodatabase schema. Developed by the Centre for Research in Water Resources at the University of Texas at Austin, the ArcGIS Hydro Data Model (Arc Hydro) is built upon the geodatabase model and provides a standardised framework that can be used to manage various types of water resource data. In particular, Arc Hydro makes extensive use of geometric networks and their analytical capabilities, and builds upon the tools available to geodatabases. Arc Hydro describes natural water systems and not constructed infrastructure, and focuses on the description of surface water hydrology and hydrography. The aim of the model has not only been to build a complete hydrological data model for use in a GIS environment, but also to create a database that can be employed by other water resource models that operate independently of GIS (Maidment, 2002; Arctur and Zeiler, 2004).

Table 2.3 The ten steps of geodatabase design (Arctur and Zeiler, 2004)

1	Identify the information products that will be produced with the GIS. Inventory map products, analytical models, database reports, web access, data flows, and enterprise requirements.	**Conceptual Design**
2	Identify the key thematic layers based on the information requirements. Specify the map use, data source, spatial representation, map scale and accuracy, and symbology and annotation.	
3	Specify scale ranges and spatial representations for thematic layers. GIS data are compiled for specific scale use; feature representation often changes between points, lines and polygons at larger scales. Rasters are sampled to include multi-resolution pyramids.	
4	Group representations into datasets. Discrete features are modelled with feature datasets, feature classes, relationships, rules and domains. Continuous data are modelled with rasters.	
5	Define the tabular database structure and behaviour for attributes. Identify attribute fields, specify values and ranges, apply subtypes to control behaviour, and model relationships.	**Logical Design**
6	Define the spatial properties of your datasets. Use networks for connected systems of feature and topologies to enforce spatial integrity and shared geometry. Set the spatial reference for the dataset.	
7	Propose a geodatabase design. Make informed decisions on applying structural elements of the geodatabase and prepare a design.	
8	Implement, prototype, review and refine the design. From the initial design, build a geodatabase and load data. Test and refine the design.	**Physical Design**
9	Design work flows for building and maintaining each layer. Each layer has distinct data sources, accuracy, currency, metadata and access. Define work flows to conform to the user.	
10	Document and design using appropriate methods. Use drawings, layer diagrams, schema diagrams and reports to communicate the data model.	

The Arc Hydro data model is what Maidment (2002) terms an essential data model, in that it captures the concepts of features, at the core of river hydrology, while leaving the user to customise it further by allowing addition to the existing structure. The goals of the model are threefold: to allow for a comprehensive hydrological description, detailed hydrological connectivity, and to provide a supporting structure for hydrological modelling. These concepts are elaborated upon by Maidment (2002):

- *Hydro Description* allows for a geospatial account of water resource features. This includes what they are, their shape, location and attributes;
- *Hydro Connectivity* provides details on how the water features are connected through the environment. Thus it can provide answers to the movement of water in a hydrological network, including what is upstream and downstream, as well as what drainage areas and water resource structures (such as gauges and dams) are connected to the network; and
- *Hydro Modelling* estimates water flow and properties for each spatial location. This involves how time-varying properties of water are described, how water resources analyses are to be carried out in ArcGIS, and the manner in which links can be established between the GIS and external water resource models.

The Arc Hydro model uses two approaches in building a geographical model, namely the inventory and behavioural approaches, which in practice are often blended together. The inventory approach is used to identify objects of interest and formally define them by recording their attributes, such as location, properties and individual behaviour. By employing the inventory approach, the formation of the *Hydro Description* (the first goal of the Arc Hydro data model) can be achieved. The behavioural approach identifies a system of behaviour for relevant features and those linked to them, thereby providing a definition for the complete system. The emphasis here is not on describing the environment in detail – this is the goal of water resources models. However, some sort of simplification is required and Maidment (2002) suggests that fundamentally, and regardless of its depiction, water resources modelling always come back to one constant, *viz.* the river. Thus the passage of water from land into a river system and beyond, allows for an orderly procedure to be established that describes the second goal of Arc Hydro, i.e. *Hydro Connectivity*.

What becomes evident upon examination of Figure 2.6, are the distinct feature and object classes in a geodatabase that allow for systematic processing. However, when this concept is

applied to the flow of water along a specific river segment within a particular catchment (while simultaneously allowing for the influence of specific entities such as lakes, canals and weirs on flow patterns), it becomes necessary to view the system in its entirety. Therefore knowledge of the connectivity between features in different layers is required. Although the geometric network includes the relationship between points and lines, it does not factor in areas, and therefore requires association with drainage data. Despite every object in an object-relational database having a unique identifier, the Arc Hydro data model has further engaged the use of an additional identifier to associate features through a *HydroID*. This is done in order to better manage the integrity of foreign keys with many relationships, since no unique identifier exists across multiple tables (Maidment, 2002).

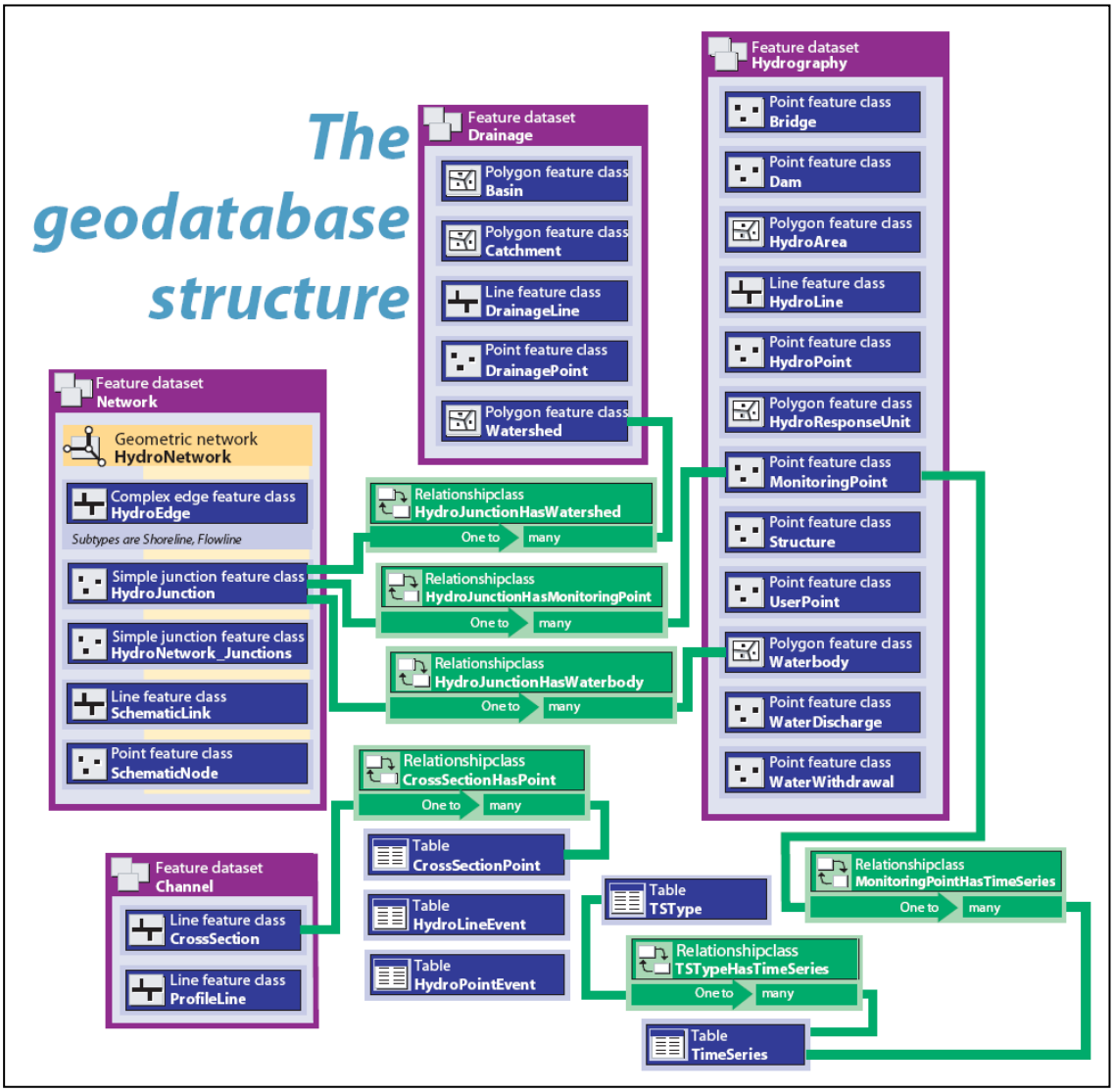


Figure 2.6 The geodatabase structure of the Arc Hydro data model, showing its various classes (ESRI, 2003b)

Thus, a hydrological feature can be linked to another by storing the *HydroID* of the first feature, but as an attribute of the second feature. A *HydroCode* is also included, that distinguishes features from one another and provides a universal identifier to be used by systems external to Arc Hydro, in order to be able to record other naming conventions that might exist. This supports the third goal of the Arc Hydro data model, i.e. *Hydro Modelling* (Maidment, 2002).

In order to support the various thematic layers used, the Arc Hydro data model organises the natural water system into five distinct components, shown in Figure 2.6. These are *Network, Drainage, Channel, Hydrography, and Time Series*. The first four are contained within individual datasets within the geodatabase, while *Time Series* is stored in a single object class (Arctur and Zeiler, 2004). The *Hydro Network* is the mainstay of the Arc Hydro data model and is made up of connected sets of points and lines presenting pathways of water flow. The network is based upon the integrated geometric network capabilities of the geodatabase, and is comprised of *HydroEdges* and *HydroJunctions*. Edges represent flow lines through streams and water bodies, as well as shorelines around water bodies. Junctions serve as connecting points at intersections of edges, and as such they are one of the most important elements of the Arc Hydro data model, which is to connect hydrographic features to the hydrological network, such as assigning a gauging weir to a network junction. With a complete hydrological network, flow direction along river segments is determined, and this allows for querying of the network, for example upstream tracing (Maidment, 2002; Arctur and Zeiler, 2004).

Another important characteristic of Arc Hydro is the derivation of drainage areas (or catchments) and stream lines from surface topography. A critical simplification of Arc Hydro is the idea that drainage areas flow to points on lines rather than directly onto lines themselves. This results in the assumption that all water within a catchment flows to its outlet, and thereby removes potential inconsistencies of internal catchment drainage. A subsequent benefit of this is the potential to substitute any drainage configuration in place of those generated by Arc Hydro, whether they are river or catchment definitions. This brings to the fore one of the useful tools made more readily available by the integration of Arc Hydro into ArcGIS. Through the use of a detailed DEM, ArcGIS is able to analyse the drainage patterns of the land surface terrain, and thereby derive its own river and catchment definitions. This is made possible by a stepwise process of determining flow direction, flow

accumulation, stream definition, stream segmentation and catchment grid delineation respectively. These tools form part of a greater collection called the *Arc Hydro Toolset.* This toolset allows for automated processing of drainage patterns that would otherwise take far longer if trying to interpret them from contour lines on maps (Maidment, 2002; Arctur and Zeiler, 2004). The aforementioned tools were also previously available in ArcInfo, and can be used in ArcGIS independent of ArcHydro. ArcHydro does, however, provide a stepwise and simpler user interface for their execution.

In some hydrological modelling the analysis is designed to work with the simple connectivity of catchments or other water-related features. Thus a simplification of the *Hydro Network* is available in Arc Hydro through the use of the Arc Hydro Toolset. The product is a schematic network of nodes and links that connects catchments according to their logical routing. The creation of a schematic network provides the connectivity between features, without the complication of their complete geometry. The tool functions by establishing a relationship between the outlet of each catchment, as well as the outlet point of the catchment downstream from it (Maidment, 2002; Arctur and Zeiler, 2004). Thus a simple network can be created that is suitable where internal drainage within a catchment is not an issue.

The third component of the Arc Hydro model is the channel, which is stored as a three dimensional shape representing the river. These shapes are divided into cross-sectional lines perpendicular to the river, and into parallel profile lines. The description of the river channel is an important part of hydrology, especially in the case of floodplain mapping and stream ecology and geomorphology. The physical properties of the channel such as roughness and vegetation can be recorded and associated with their particular reach, where sufficient data have been collected. The fourth and final of the individual datasets is the hydrography, which holds base data collected from topographical maps and other sources. Multiple layers can be stored in this dataset, and they are grouped according to point, line and area classes. Points are further subclassed into dams, bridges and structures that do not alter flow, and water withdrawal and discharge points along the *Hydro Network*. Two additional point feature classes are defined, *viz.* monitoring points and user points (Maidment, 2002; Arctur and Zeiler, 2004).

Since many observations important to hydrological modelling are taken at regular time intervals, as is the case with daily rainfall, it is necessary to be able to incorporate these data

into a database (Wachowicz, 1999). It has in the past been difficult to include time series data with other GIS data, since GIS data models do not normally consider temporal information. Thus the Arc Hydro data model includes an object class for time series which is stored as tabular attribute data and describes the time varying water properties of a feature. It should be noted that the time series information stored does not have to be exclusive to features associated with that hydrological network, and can include data from rainfall gauges and other disjunct features (Maidment, 2002; Arctur and Zeiler, 2004). Additional problems arise, however, as a result of the large size of temporal databases and inefficiency of storage. More detail on the time series component of Arc Hydro is provided in Section 3.3.

In order to meet particular user requirements, application or industry-specific data models such as Arc Hydro have also been developed with the geodatabase and are, in a sense, extensions of it. In addition to its various hydrological components, the Arc Hydro data model has provided one means by which to store voluminous time series data. The storage of time series data forms the basis for the following chapter, which reviews a number of potential time series data storage methods.

3. STORAGE OF TIME SERIES

The previous chapter explains why the geodatabase data model was selected to store spatial data, and why the Arc Hydro data model was selected for the storage of hydrologically related features. One of the other requirements of the data model was that it be able to store time series data. Time series data is of particular importance in hydrological modelling as opposed to just spatial data in many other GIS applications. The choice of which time series storage method to use was important for the requirements of the new data model as listed in the Introduction, since data output from or input into hydrological models is often made up of a significant amount of time series data.

A time series may be defined as a set of ordered observations on a quantitative characteristic of an individual, or collective, phenomenon recorded at different points in time (NCES, 2006). Stated simply, it is a set of observations that are arranged chronologically (Hipel and McLeod, 1994). Time series data describe many components of the hydrological cycle and they are often used as an input for hydrological analyses and modelling. The two main factors that determine the use of time series data are the length of the time series record and its frequency. Frequency refers to the time interval between recordings, and may differ in relation to requirements, such that during a flood, frequency of recordings might be increased in order to improve the detail of the event (Maidment, 2002).

Time series may be discrete (for example, recording only the biggest event of a year, or the 'n' biggest events) or continuous, in which case they may be recorded at regular intervals (such as every 10 minutes or daily) or at irregular intervals (for example, when change takes place, as in a flood hydrograph). In hydrology, time series originate in one of two ways: actual observed data or values generated by models. It is important to know the origin of values in a time series in order to distinguish actual measurements (or data) from simulations (or values). The association of time series data with spatial objects (and therefore GIS), results in spatio-temporally varying hydrological variables such as rainfall and temperature being modelled and recorded more accurately, thus resembling their occurrence in reality more closely. There are a number of properties associated with the recording of time series data. These are its accuracy, duration and whether the time series has a regular or irregular interval. Time series values are also commonly distinguished as instantaneous or step values.

Instantaneous values represent a condition at any given instant (such as the temperature at 08:00), while step values represent a periodic or step statistic (illustrated by daily accumulated rainfall). One problem in dealing with time series data, results from the fact that these data are often voluminous, and as a result, storing and accessing them can be complicated. In the past, standard databases have been inefficient in storing and retrieving time series data, and for that reason many hydrological data models have included various designs of formatted data files to circumvent this problem (Maidment, 2002).

Since it would influence the design of the new data model, it was necessary to first select the method of time series storage to be used. The times series storage method used would not necessarily be a new design, and instead would draw on some of the methods currently available. By having a defined data model for the storage of time series, it would be possible to consider its integration with the remainder of the new data model to be developed. The data used for the evaluation of the different time series storage methods investigated, was sourced from the QCDB; discussed further in Section 3.2.

The following time series storage methods were identified, reviewed and subsequently evaluated:

- Relational database tables (two methods);
- Arc Hydro time series;
- SPATSIM time series data table; and
- DHI's dsf0 format.

These time series storage methods present different ways by which time series data can be recorded and stored, and constitute a sample of the many potential time series storage methods available at the time the study was undertaken. Those listed above were chosen from a preliminary investigation into common methods of time series storage in a GIS, as well as from peer recommendation.

3.1 Selection Criteria of the Time Series Storage Method

In this chapter, several time series storage methods are evaluated in order to determine which method would be best suited for implementation in the new data model. The time series storage methods were evaluated based on the following requirements:

- The time series storage method chosen needed to be compatible with the geodatabase, with preference to a method that was able to be stored within a Microsoft Access database, thereby allowing for a single collective database, when considering the other data to be included in the project;
- The time series method chosen was required to record the Station ID of the associated time series, in order to be able to define the original location where the time series was recorded, by relating it to a feature in GIS;
- The method of time series storage needed to be able to store discrete and continuous data from observations, or that generated by simulations that could be used for hydrological analyses and modelling;
- The precision in the storage of time series data at a minimum needed to enable storage of a large values (up to 999 999 999), including a single decimal point, while a method which could store greater accuracies of up to 10 decimal points would be beneficial;
- The potential storage of data quality flags associated with a particular set of records; and
- The method of time series storage needed to have efficient data storage since time series datasets are often voluminous.

The storage space efficiency (disk space used) was one of the main factors considered when comparing the different time series storage methods. This was important, since from early in the study it was noted that database sizes could quickly exceed 2GB. This 2GB limit is a constraining factor in that it is the maximum file size of the FAT16 (File Allocation Table). A file system is a method of storing and organising computer files, and FAT16 is a popular file system in that it supports all versions of Microsoft Windows operating system, DOS (Disk Operating System), and some UNIX operating systems. Other file systems such as FAT32 and NTFS are more limited by the operating systems with which they can be used. The 2GB limit is also applicable to the Microsoft Access RDBMS which has been designed to limit its database size to the FAT16 limit of 2GB (Microsoft, 2007).

Other factors in the choice of time series storage method included ease of use and the potential cost of purchasing software or rights for the implementation of the chosen method. The time it took to import and export the time series data into the various formats was not evaluated as it is not possible to adequately measure, in that it requires the quantification of the time it takes to run through the tasks involved in the import and export of data into each of the time series storage methods. Thus, it is not only a record of computational time, but also the physical time it takes to setup these processes. With added familiarity and more efficient computation (in the form of tailored programming code), a marked difference in the time it takes to import or export data may be seen for a particular time series storage method.

3.2 Data Used in the Evaluation of Time Series Storage Methods

South Africa, Lesotho and Swaziland have been delineated into 1 946 Quaternary Catchments that act as the smallest spatial unit of division for the region's catchments. A database of historical climate data, forming part of the Quaternary Catchments Database, was selected as the source of the time series data for the evaluation of the selected time series storage methods. The database would also be used in the implementation phase of this study, reviewed in Chapter 5. This database (Hallows *et al.,* 2004; Schulze *et al.*, 2005; Schulze *et al.,* 2006) was developed over the past 10 years by the School of Bioresources Engineering and Environmental Hydrology (BEEH) at the University of KwaZulu-Natal, for use with, *inter alia,* the ACRU agrohydrological simulation model (Schulze, 1995). The QCDB comprises daily time series data, and includes rainfall and maximum and minimum temperature. The time series data were originally stored in text files in the ACRU composite format (Smithers *et al.*, 1995) where one text file was used to store the time series data for each of the 1 946 Quaternary Catchments. These composite files cater for a wide selection of daily input data and have a fixed file format. Each text file was approximately 657 KB in size, resulting in a total dataset size of roughly 1 280 MB. Every file had a total of 18 262 lines, representing a complete discrete record of 50 years from 01/01/1950 to 31/12/1999.

A sample of an ACRU composite format time series file is shown in Figure 3.1, while Table 3.1 provides a description of this format. A problem with the original format of storage was the large amount of data replication. This was as a result of there being one historical climate file for each of the region's Quaternary Catchments, and since rainfall data were unique to

1 244 rainfall stations used in the 1 946 Quaternary Catchments, the rainfall time series data were duplicated in some of the Quaternary Catchments climate files. Temperature data on the other hand, were specific to each of the Quaternary Catchments, and therefore no data replication occurred.

```
0674100W19990727  0.0  23.6   7.7
0674100W19990728  0.0  23.2   6.2
0674100W19990729  0.0  24.6   8.0
0674100W19990730  0.0  26.0   8.9
0674100W19990731  0.0P 26.6   8.7
0674100W19990801  0.0  28.3   9.9
0674100W19990802  0.0  29.5   9.7
```

Figure 3.1 Sample from an ACRU composite format time series file

Table 3.1 Description of the format of an ACRU composite file (Smithers *et al.,* 1995)

Characters	Description
1 – 8	Specifies the ID of the monitoring stations
9 – 16	Daily date stamp of the record
17 – 21	Specifies the rainfall amount in mm/day
22	A quality code
23 – 27	Maximum temperature in degrees Celsius
28	A quality code
29 – 33	Minimum temperature in degrees Celsius
34	A quality code

For the purpose of comparing the various methods of time series storage that were selected, it was decided that historical climate data from 500 Quaternary Catchments would be used as a sample of the complete historical climate files database. The exclusion of approximately three quarters of the remainder of the database was done in order to fall within the 2GB limit of Microsoft Access, and speed import and export of data. A quality code was included in the composite format that represents the status of the recorded time series data, such that in the case of rainfall data, missing or incorrect values have been synthetically derived and infilled in order to improve the integrity of the time series. The quality code was excluded, however, from the evaluations of time series storage methods, since the quality code was not continuous and was incompatible with some of the methods that were to be evaluated. The

dataset size of the selected ACRU composite files, including rainfall and temperature data, was 329 MB.

3.2 Relational Database Tables

Two approaches of storing the time series in a relational database tables were investigated. The first used simple database tables whereby data was split into two tables; one for temperature and one for rainfall, with Station IDs and dates being recorded for each record in both tables. The second approach utilised a single dynamic database table in a database, in which all the time series data was stored. A second table was then used to capture attribute data, including the Station ID, and the relative period of record of the time series.

3.3.1 Simple database tables

The purpose of the first of the two approaches was to import the ACRU composite files into a relational database with two tables, one for rainfall data and another for temperature data. The reason for the utilisation of two separate tables was to maintain the naming conventions used in the original ACRU composite files, since rainfall data for a particular Quaternary Catchment was uniquely identified by an internal station name held within the ACRU composite file, while temperature data for a Quaternary Catchment was uniquely identified by the external file name of the associated ACRU composite file. Although possible to produce, the use of a single database table for the storage of both rainfall and temperature data, would have resulted in the replication of rainfall data, had the complete historical climate dataset been used. As a result of the aforementioned replication of rainfall data in some of the ACRU composite files, it was decided to avoid unnecessary complication in subsequent data dissemination, by storing the rainfall and temperature data separately. The creation of simple database tables would also allow for a new option of time series storage to be researched, as well as providing a necessary link in the application of the Arc Hydro time series storage method. Microsoft Access was selected as the RDBMS in which the two tables (for temperature and rainfall) would be created and populated. The decision to use this particular RDBMS was its availability, as well as its suitability in terms of the geodatabase. Thus, two tables were created in the RDBMS with attribute fields for station name and time stamp, as shown by Tables 3.2 and 3.3. For the temperature database, additional fields for daily maximum and minimum temperature were added, while the rainfall database included a field

for daily rainfall. Both database table values for temperature and rainfall were stored as *single* data types. The relevant Microsoft Access data types are listed in Table 3.4.

Table 3.2 An example of the simple table used to store rainfall data

Field	Description	Data Type	Example
ID	A unique number generated automatically	Auto Number	180
StationName	Stores the station name	Text	0087635W
Time_Stamp	Used to record the date of the record	Date/Time	1950/07/01
Rain	Daily total rainfall amounts	Number (Single)	4.6

Table 3.3 An example of the simple table used to store temperature data

Field	Description	Data Type	Example
ID	A unique number generated automatically	Auto Number	180
StationName	Stores the station name	Number (Long Integer)	233
Time_Stamp	Used to record the date of the record	Date/Time	1950/07/01
MaxTemp	Daily maximum temperature records	Number (Single)	24.0
MinTemp	Daily minimum temperature records	Number (Single)	5.1

The final size of the database was 497 MB. This size, as with all the relational databases created during this study, is not exactly accurate in presenting the true size of the time series data stored by the database, as some database overhead is required for other information used by Microsoft Access to maintain database integrity. This is illustrated by an empty Access database which has a file size of 96 KB. Although easy to manage, with simple export and import options, a problem with this time series storage method was the large size of the database. There was some data replication since the time stamp of each record needed to be captured twice, once for each of the two database tables. Additionally, each record in the database required an *ID* as well as a value in the *StationName* field, which added to the storage space required. An important point to note is that no database indexing was used for any of the time series storage methods reviewed, since this would potentially result in an incorrect comparison of disk space used by the various methods.

Table 3.4 The various data types used in Microsoft Access tables, and their sizes (Databasedev.co.uk, 2006)

Data Type	Storage Size
Auto Number	4 bytes
Text	Variable
Integer	2 bytes
Long Integer	4 bytes
Single	4 bytes
Double	8 bytes
Date/Time	8 bytes

3.3.2 Dynamic database table

The second approach made use of a slightly more complicated, but more efficient database structure. The data held in the tables created in 3.3.1 above, was imported into a single field of a new database table; TimeSeries_Data. Only two fields were present in this new table; an autonumber (*ID*), and a time series value field (*data*). Rainfall, maximum and minimum temperature values were therefore combined together into a single field, with no identifiers to differentiate between the two variables stored within the table. This differentiation was, however, achieved through the use of a second table which stored time series attribute information. This TimeSeries_Attributes table recorded the position of a particular time series in the TimeSeries_Data table, by capturing the start record and end record for a particular time series, along with the *StationName*, and *TSType* (rainfall, maximum or minimum temperature). The number of records, start date and end date for each time series were also recorded. Thus, it was possible to store the time series effectively as it had a fixed interval with a known start, end date and number of records. This approach draws on the method used by SPATSIM in Section 3.5 below, in that it subdivides time series data into time series data values, and supporting attribute information. Tables 3.5 and 3.6 define the structure of the tables.

The final size of the database was 220 MB. The advantage of using an efficient time series storage method is demonstrated by this time series storage method having halved the size of the database described in Section 3.3.1. This reduction in size was due to the absence of unnecessary data replication. A disadvantage of this storage method is that due to the large amount of data stored consecutively in the *TimeSeries_Data* table, extra care needed to be taken to accurately record the time series attribute information, since date values are not

recorded for each data value. This time series storage method has the limitation of only being suitable for fixed interval time series.

Table 3.5 An example of the *TimeSeries_Attributes* table used to store attribute time series information.

Field	Description	Data Type	Example
ID	A unique number generated automatically	Auto Number	315
StationName	Stores the station name related to the raingauge or temperature recording station	Number (Long Integer)	0087635W
TSType	Records the type of time series data being stored.	Text	Rainfall
Start Date	Date at which the time series starts	Date/Time	1950/01/01
End Date	Date at which the time series ends	Date/Time	1999/12/31
No. of Records	Number of records contained in the time series	Number (Long Integer)	18262
Start_Record	ID of the record in the Time Series table at which the time series for a particular TS_Type and StationName starts	Number (Long Integer)	219144
End_Record	ID of the record in the Time Series table at which the time series for a particular TS_Type and StationName ends	Number (Long Integer)	237406

Table 3.6 An example of the *TimeSeries_Data* table used for the storage of time series data.

Field	Description	Data Type	Example
ID	A unique number generated automatically	Auto Number	913100
Data	Value of a single record in the time series	Number (Single)	4.6

3.4 Arc Hydro Time Series

A brief summary of the Arc Hydro data model's time series component was provided in Section 2.4. The purpose of including time series data in the Arc Hydro data model is not only to build a complete hydrological data model, but also to create a data model that is accessible to water resource models that operate independently of a GIS. Integration of these models with GIS can be addressed by knowing how the time series data are stored and by knowing the format of the hydrological model input files.

Time series data that are directly stored in an Arc Hydro geodatabase are represented in a table known as a Time Series Object Class. The time series data in an Arc Hydro geodatabase are essentially an instance of a relational database table. Four particular fields are required for

the storage of time series data in the Time Series Object Class, namely *FeatureID*, *TSType*, *TSDateTime* and *TSValue,* as indicated by Figure 3.2. Maidment (2002) further points out that the structure of time series data storage in Arc Hydro is such that all the time series data are stored in one database table, regardless of its feature type (such as a rainfall monitoring point or gauging weir) and the type of data stored in it (rainfall or streamflow respectively).

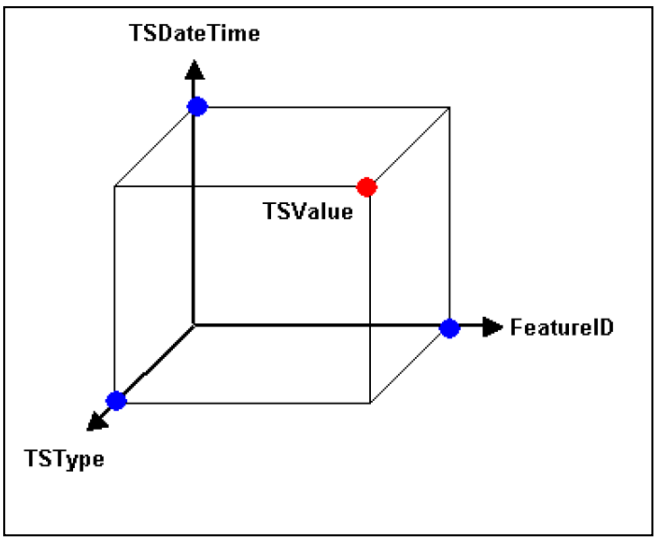


Figure 3.2 Three dimensions of the *TimeSeries* object class (Maidment, 2002)

In Arc Hydro, the Time Series Object Class manages the association between a spatial feature and its multiple time series measurement attributes. Thus, when considering Figure 3.2, it is possible that a table holding time series data may have many recorded values for the same *FeatureID*, *TSType* or even *TSDateTime* when viewed individually. However, when all three measurement attributes are considered, only one value is possible. Figure 3.3 shows the specific model structure for time series in Arc Hydro. In this figure, table *TimeSeries* is treated as a non-spatial table of records, with a *FeatureID* field that links to the associated feature in a separate feature class *MonitoringPoints,* as in Figure 2.6. This *FeatureID* has a cardinality of one-to-many and means that one feature (in feature class *MonitoringPoints*) can be linked to any number of time series objects (in the *Time Series* table). A second database table important to the Arc Hydro time series storage method is that of *TSType*. This table is in turn, linked to the *TSType* field in the *TimeSeries* object class and provides attribute information particular to the associated record. The *TSType* table in Figure 3.3 provides details on the attributes recorded (Maidment, 2002).

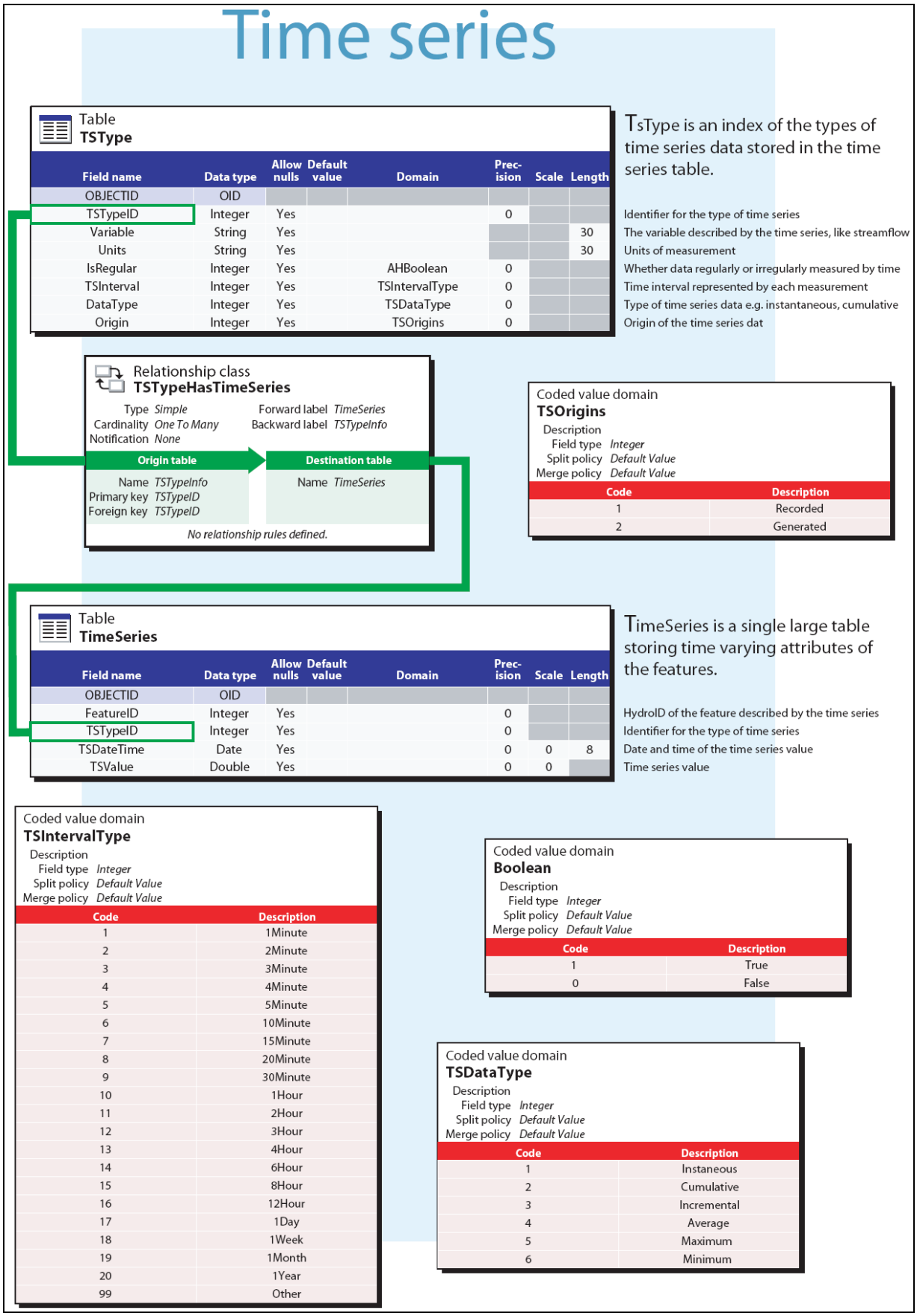


Figure 3.3 The Arc Hydro data model for storing time series (ESRI, 2003b)

A database founded on the Arc Hydro data model's *Time Series* object class was created in Microsoft Access. The previously created simple database tables were used as the data source for this process. The results obtained from the Arc Hydro time series database produced a significantly larger database size of 802 MB when compared to the simple database table method. This large database size is as a result of a table record being created for each individual time series value (such as maximum temperature), while the simple database table method stores maximum and minimum temperature as a data pair, in the same record. Thus there are twice the number of temperature records in the Arc Hydro time series table as there are in the simple database table. In the case of the Arc Hydro time series storage method, additional disk space is being used for the capturing of the associated *TSDateTime*, *TSType* and *FeatureID* information. Although the Arc Hydro time series database used the *single* data type for storage of its values, a second Arc Hydro database was created using the *double* data type. This second table was created because the dfs0 format method, discussed in Section 3.5, required the use of an Arc Hydro database with time series values stored as *doubles*, for future importing purposes. The final size for this second Arc Hydro database using a *double* data type was 941 MB. The Arc Hydro method nonetheless provides a structured and clear method by which to store time series data.

3.5 SPATSIM Time Series Data Table

The Spatial And Time Series Information Modelling (SPATSIM) software package is described by IWR (2004) as an integrated hydrology and water resources information management modelling system. SPATSIM was developed by the Institute for Water Research (IWR) at Rhodes University, and is based on a previous hydrological model application system called HYMAS, which was developed by IWR during the late 1980s and early 1990s. Although proving successful for applying a variety of hydrological and water resource models, HYMAS had some major disadvantages, mainly in regard to its methods of information storage, and its lack of a spatial reference component. A large part of the new system design in SPATSIM was inspired by the approach adopted by the Centre for Ecology and Hydrology at Wallingford, UK (Hughes and Forsyth, 2002).

The four main components in the design of SPATSIM are the spatial interface, the main database structure, the internal utilities and the generic approach to accessing external utilities and models. SPATSIM's spatial interface is realised through ESRI's MapObjects, with

shapefiles accounting for most of the spatial information. Additional attribute information contained by the shapefiles are not used in the original format, but are rather transferred to a SPATSIM database. This is as a result of the need for consistency in data types, as well as the inability for shapefiles to store time series data (Hughes and Forsyth, 2002).

The main database structure of SPATSIM is shown in Figure 3.4 and consists of shapefile tables, four data dictionary tables and attribute data tables used to store attribute data associated with each spatial element (point, line or polygon) in a shapefile (Hughes and Forsyth, 2002). Each feature class is stored in an individual shapefile. The first of the data dictionaries supplies a list of features, which provide a link to the spatial data stored in the shapefiles. The second data dictionary uniquely associates attributes with a feature defined in data dictionary 1, while the third data dictionary provides the link between the list of attributes in data dictionary 2 and the database attribute data tables in which the attribute data are stored. The fourth data dictionary links the attribute data table codes from data dictionary 3 with the unique record IDs in the shapefiles.

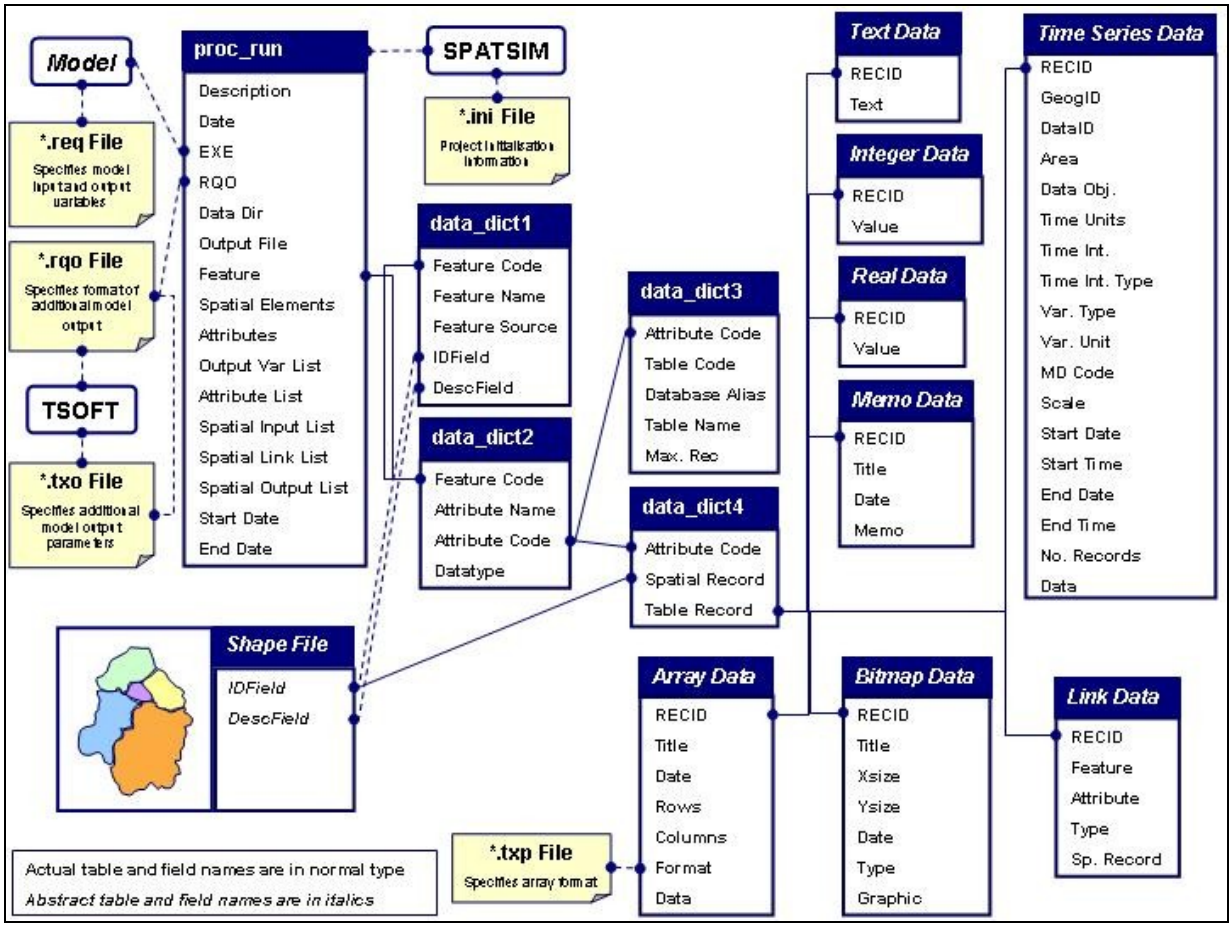


Figure 3.4 Structure of the SPATSIM modelling system (Clark, 2005)

There are eight attribute type tables to account for the storage of data types commonly used in water resource modelling. These include the following (Hughes and Forsyth, 2002):

- *Text*;
- *Integer*;
- *Real* for storing floating point values;
- *Bitmap* for storing images or videos;
- *Array* (data matrices);
- *Memo* for holding any metadata;
- *Link* for creating an association between an attribute and a different feature; and
- *Time Series* –for storage of SPATSIM's format of time series.

SPATSIM uses a specific format to try and maximise the efficiency of time series storage and thereby minimise the amount of disk space required. This is achieved through the incorporation of a number of fields in the Time Series attribute data table that are used in an attempt to prevent any unnecessary replication of data as shown in the time series data object in Figure 3.4. A description of each of the fields in a time series attribute data table in SPATSIM is provided in Table 3.7. The actual time series (observed data or generated values) for each time series of at particular location and each type of data (such as rainfall or temperature), are stored within a Binary Large Object (BLOB) in the *Data* field, as a single record in a time series database table. When the time series data being stored have a regular interval, the values are stored in the BLOB as consecutive floating point decimal numbers represented using 4 bytes. Thus through the interpretation of the start and end date/time, and the number of records, it is possible to reconstruct a table containing time and value pairs. Other information pertinent to a time series can be inferred from the values contained in the attribute fields, such as the units of measure, missing date code and metadata about the time series. When the time series has an irregular interval of recording, an additional date/time value is stored for each data value in the BLOB.

The time series storage method of SPATSIM was replicated in a Microsoft Access database. This required that a table be created in Microsoft Access with the fields as listed in Table 3.7. For the creation of the SPATSIM database, each of the maximum temperature, minimum temperature and rainfall values were captured in separate BLOBs in the *Data* field of the table. No other data other than the actual recorded values of these variables were present in

the BLOBs. This was because the imported time series data had a regular interval and thus no time stamp was required to be recorded, as this could be inferred through the other time series attribute data table fields. The SPATSIM time series storage method does, however, allow for the inclusion of time stamps associated with discrete TS data.

Table 3.7 The format of the SPATSIM time series data table when used in Microsoft Access (Hughes and Forsyth, 2002)

Field	Description	Data Type	Example
RECID	Table record field of data dictionary 4	Auto Number	66
GeogID	Up to 80 character description of the location of the time series data	Text	0087635W
DataID	Up to 80 character description of the type of data stored	Text	Rainfall
Area	Catchment area if required (km^2)	Number (Single)	3451.85
Data Obj.	Identifies regular or irregular time step of time series data	Number (Integer)	0
Time Units	Units of time data; 1=minutes; 2=hours; 3=days; 4=months; 5=years	Number (Integer)	3
Time Int.	Time interval (integer) when the data regular time step	Number (Integer)	1
Time Int. Type	Time interval type when the data object type = 1. Intervals of recording or cumulative.	Number (Integer)	0
Var. Type	Variable type of the time series data E.g. 1=length, 2=area	Number (Integer)	1
Var. Unit	Variable units type of the time series data. 1=Litres, $2=m^3$, etc.	Number (Integer)	5
MD Code	The code to be used to recognise missing data	Number (Single)	-9999
Scale	A field used to scale the data to ensure that it corresponds to the units specified in Var. Unit	Number (Single)	0
Start Date	Start date of the time series data	Date/Time	1950/01/01
Start Time	Start time of the time series data	Date/Time	12:00:00 PM
End Date	End date of the time series data	Date/Time	1999/12/31
End Time	End time of the time series data	Date/Time	12:00:00 PM
No. Records	Number of records in the time series data	Number (Integer)	18262
Data	BLOB of the time series values	OLE Object	Long Binary Data

The use of the SPATSIM time series storage method resulted in a database size of 111 MB, as a result of the time series values in the BLOB being stored as *single* data types. This database contained no data replication and maximised the database efficiency by using a relatively small amount of disk space. A refinement of this time series storage method was evaluated by creating a new database using an *integer* data type for the temperature and rainfall values, in

place of the original *single* data type. This resulted in database size of 53 MB, approximately half of the original. This was attributed to the fact that an integer data type value has a size of 2 bytes, whereas a *single* data type is 4 bytes. In order to preserve the integrity of the database, the original climate values were multiplied by a factor of 10, thereby removing the previously necessary decimal place, since there was a maximum of one decimal place for all the original climate values. This change was captured by the *Scale* field in the database in order to allow for the necessary alterations to be made when extracting the time series data.

It should be noted that the conversion of single data type into integer data types is also possible for all of the other time series storage methods described in this chapter. This conversion was not done for the other time series storage methods, as it would have resulted in duplication of datasets, with little change in the manner of storage. It was, however, implemented in the SPATSIM database since it was an option inherent to this time series storage method, and allowed for the benefits of the conversion to be investigated.

A disadvantage of saving data in integer form, however, is that although it is appropriate for the storage of rainfall and temperature data used for evaluation purposes, it is not suitable for time series data requiring more precision. Consequnetly, it is not possible to accurately store large data values of over 32 768 or data values with numerous decimal points using a 2-byte integer.

The SPATSIM time series storage method resulted in a small database, which required specific Visual Basic code in order to facilitate the importing of time series data from the simple database tables into each BLOB. This is because the data stored in the BLOB is not stored in a user readable form, and requires that it first be exported into a secondary format before it can be reviewed or used.

3.6 DHI's dfs0 Format

Developed by the Danish Hydrological Institute (DHI), the data file system (dfs) format is a proprietary format developed specifically for the storage of geographic data, including time series data. The dfs format is a general file format used by DHI, and as such has a structure generic enough to support all data types required by DHI's suite of models. The dfs format can be subdivided according to the number of spatial axes involved. Thus, where there is no

spatial axis, a file suffix of dfs+0 would be given, while where there is one spatial axis a suffix of dfs+1 would be required, and so on, with a final declination of dfs+U being given to data stored as an unstructured grid. Time series points would therefore have the file extension dfs0, lines would be dfs1, grids would be dfs2, and 3-D grids would be given the extension dfs3. There is, however, no limitation to the types of data that could be stored within a single dfs file, and both time series and gridded data can be found together in the same dfs file (Bech, 2005).

The general file structure of the dfs format is shown in Figure 3.5 and is comprised of four main components. The first component of the file is that of the necessary identification tags, and classifies the type of data stored with regards to it having a combination of a regular or irregular time and spatial axes. The second component is the header, which provides a list of items in the file and acts like a table of contents. The header itself is further divided into three parts made up of *Global Parameters*, *Primary Axis* and *Item Description*.

The *Global Parameters* part of the header contains basic information required to be able to read and interpret the data correctly. These parameters are static information that applies to the entire file. They include information about delete values, floating point formats, topography and geographic information. The *Primary Axis* part of the header specifies properties of the time series, such as the number of intervals, the length of intervals, the units and whether the time series has a time stamp with a regular interval or not. In the case that the time series does not have a regular interval, then each step in data block (which stores the actual time series values) will have an initial primary axis value, such that its time stamp can be recorded. The final part of the header is that of the *Item Description*, which although similar to the primary axis contains additional information such as a textual description of the data. The third component is the static vectors or static items, and is used to store data which does not vary in time, such as elevation data used as input for DHI's MIKE 21 model (DHI, 2006). The last component is the data block which consists of *N* number of steps (*N* can be found in the *Primary Axis* part of the header) which contains the actual recorded time series data. Different subsections in the file structure, indicated by a small grey rectangle in Figure 3.5, are used to record the position in the binary file of each subsection. A subsequent part of the dfs format is that of the binary implementation of data storage, whereby imported data are recorded. Within the dfs format, a number of different variable types are supported, as shown in Table 3.8 (Bech, 2005).

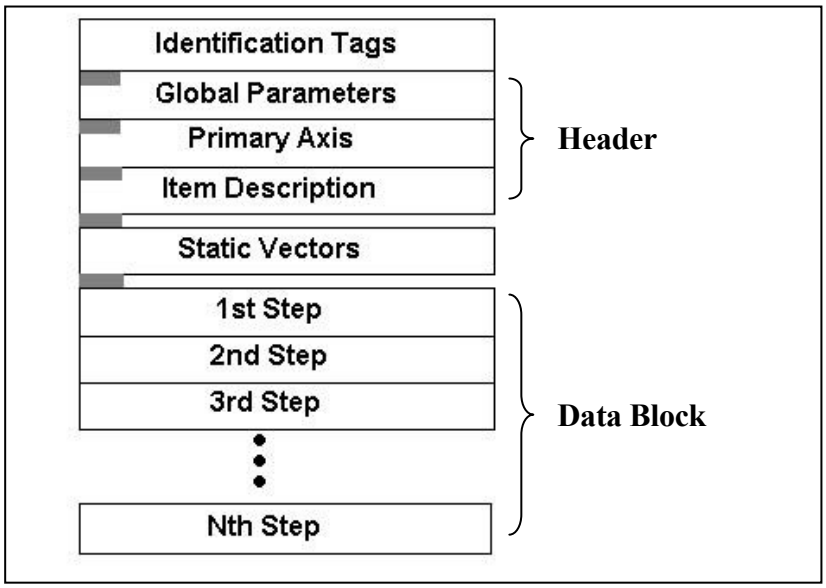


Figure 3.5 The file structure of the dfs format (Bech, 2005)

Table 3.8 The various data types used in the dfs format, and their sizes (Bech, 2005)

Data Type	Storage Size
Float	4 bytes
Double	8 bytes
Character	1 byte
Integer	4 bytes
Unsigned Integer	4 bytes

There were two possible procedures whereby the data could be exported into the dfs0 format, either through the use of a supplied Microsoft Excel spreadsheet with a built in macro, or through the Arc Hydro database. For this study it was decided to use the Arc Hydro method, as Microsoft Excel was limited by the amount of data it could store. In order to proceed with the Arc Hydro method of importing data into the dfs0 format, it was necessary to use the Arc Hydro time series database using *doubles*, as detailed in Section 3.3, as this was a prerequisite to importing the time series data into the dfs0 format. It was then necessary to use DHI's Temporal Analyst software, which is an extension of ESRI's ArcMap software, as shown in Figure 3.6. The time series data in the Arc Hydro database was imported into Temporal Analyst and subsequently exported into the dfs0 format.

The results of the dfs0 format of time series storage were surprising when compared to the Arc Hydro database from which the data were imported. The dfs0 format resulted in a

database size of 357 MB compared with 941 MB of the Arc Hydro time series storage method when using a *double* data type. The dfs0 format was also highly compatible with Temporal Analyst as a result of its supported format, and thus could take advantage of the tools provided by this ArcMap extension. This method was a particularly difficult one to implement when compared to the other time series storage methods researched, owing to the complications involved in creating a suitable database from which to import data. Since a particular format for the Arc Hydro Time Series database was required, unnecessary space was used by importing the time series data as a *double* data type, and a smaller database would have resulted had the use of a *single* data type been possible. An additional factor to consider for the dfs0 time series storage method is the purchase cost of the software.

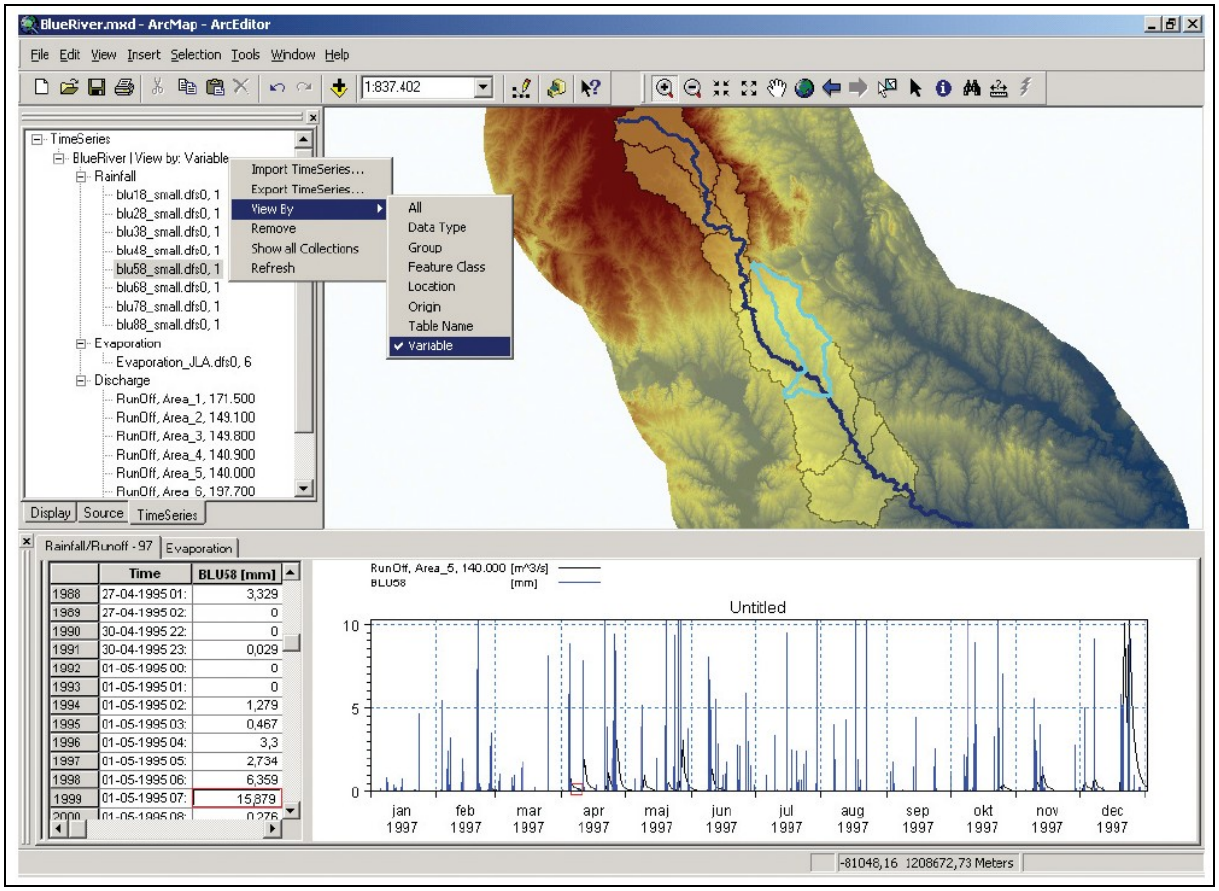


Figure 3.6 A screen capture of Temporal Analyst's integration with ArcMap (Arctur and Zeiler, 2004)

3.7 Comparison of Time Series Storage Methods

The criteria used to evaluate the time series storage methods investigated are listed in Section 3.1. All the time series storage methods investigated were able to store the Station ID of the associated time series, however, it was the original ACRU composite file format that was least compatible with the geodatabase, and would have required the creation of specific visual basic code to enable its integration. The original historical climate dataset made up of ACRU composite files had a simple format of time series storage that found its main strength in the fact that it was already compatible with the particular model for which it was created. In terms of further use, ACRU composite file storage method is not suitable for use in other hydrological models and requires processing to convert the data to other storage formats. In addition, this format results in a large amount of data replication when the entire database is considered, as it duplicates rainfall data in some of the composite files. Finally, a large amount of disk space is required to store time series data in text files such as these. Figure 3.7 illustrates the amount of disk space used by each of the time series storage methods investigated.

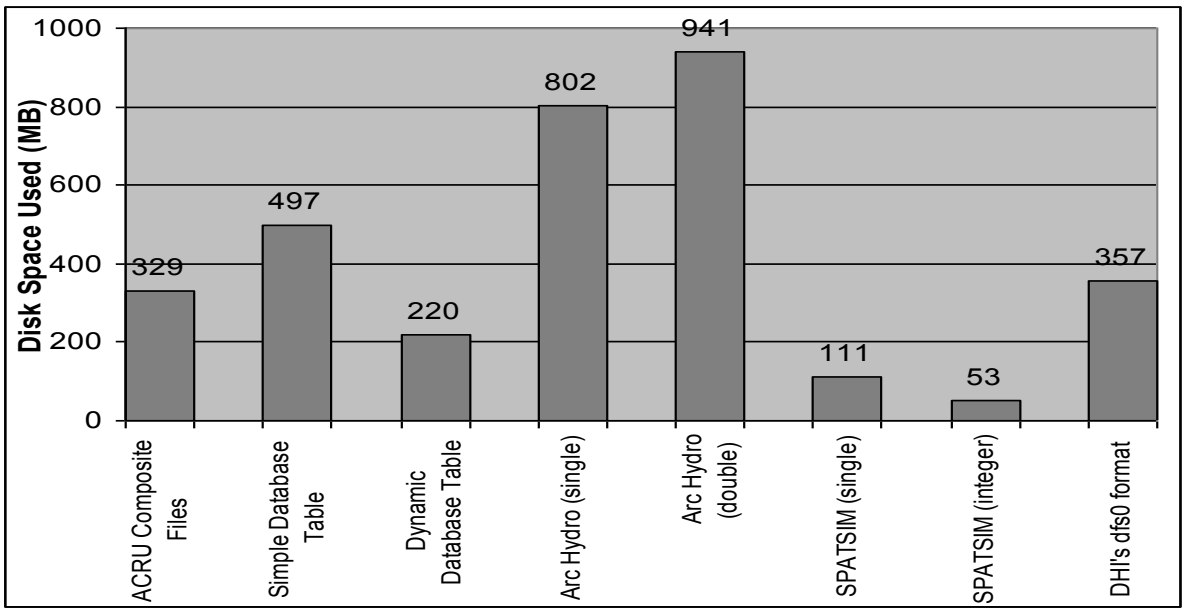


Figure 3.7 Comparison of the time series storage methods investigated

The second time series storage method researched was that of relational database tables implemented in Microsoft Access. Two approaches were investigated. The first was the storage of time series data in simple database tables. This storage method proved to be a valuable format for the exportation of time series data to the other storage methods researched and, additionally, was relatively easy to manage. The main disadvantage with this format, however, was that it still resulted in a considerable database size including significant data replication. The use of a dynamic table of time series data greatly reduced data replication of the simple database. This method produced an efficient database, with the overall database size being small, at approximately half the size of the simple database tables. This method, however, was only suitable for the storage of a continuous time series, as the timestamp of the time series value was not recorded. Discrete time series data could therefore not be stored using the method.

The third option of time series storage researched was the Arc Hydro time series database. Although the database had a logical structure, the final database proved unsuitable as a result of its large database size. This database size was a result of Arc Hydro using a single table record for each independent time series observation as well as for each climatic variable of rainfall and maximum and minimum temperature, which all required a *TSDateTime*, *TSType* and *FeatureID* value. The structure of the time series component of the Arc Hydro data model was, however, effective in that the attributes of each time series were only recorded once in a *TSType* table. This storage of attribute data is significant as it captures information about the time series data being stored, such as the origin of the data and its units of measure. The Arc Hydro time series storage method was thereby similar to that of the SPATSIM method in terms of the recording of attribute information.

SPATSIM created a database that had an efficient and logical design which allowed for an extensible database structure and little data replication. Moreover, the database size, when using either the option of a *single* or *integer* data type, was the smallest of the time series storage methods researched. The primary disadvantages of this method was that in order to view a time series, it was first necessary to extract the data from the BLOB in which it is stored, and the method was one of the most difficult to implement. The SPATSIM time series storage method could, however, benefit from the use of a table similar to the *TSType* table in the Arc Hydro, as it would allow for reduced data replication with respect to attribute information.

The final time series storage method researched was that of DHI's dfs0 format. This format produced good results, considering it used time series values stored as *doubles* for its input. The dfs0 format was also highly compatible with Temporal Analyst and other software distributed by DHI, and was the only option to include immediate visualisation and simple analysis of time series data, as illustrated by Figure 3.6. This time series storage method was the only method evaluated which would not allow for the recording of data quality flags. Another disadvantage of this format was the indirect approach required to place the time series data into the dfs0 format. Another disadvantage was the high purchase cost of the Temporal Analyst software, which is required for the importing process. A final concern of this format was that, since it was a proprietary format, there was little documentation available to describe the format of the dfs0 files.

In summary (with reference to the evaluation criteria), the ACRU composite file format was not compatible with the geodatabase without the addition of external code. The simple database table produced a database of a substantial size, while the composite file format did not allow for the storage of discrete time series data. Both the *single* and *double* data type implementation of the Arc Hydro method produced large databases, making them unsuitable for use in the project, while DHI's dfs0 format was unable to store data quality flags.

Based on the evaluation into time series storage methods, it was decided that the SPATSIM method was the most suitable for use in the remainder of this study, as a result of its excellent disk space efficiency and the logical design of the database structure, while Arc Hydro presented design elements that could potentially improve the final implementation of a time series storage method in this study.

This concluded the evaluation of the various time series storage methods investigated, and thus the following chapter will focus on the design of a geographic data model for the storage of data related to hydrological modelling.

4. DATA MODEL DESIGN

The objective of this chapter is to describe how a new hydrological data model was designed using the concepts and results described in Chapters 2 and 3 of this document. The emphasis of the design was on the creation of a new data model that enabled the storage of spatio-temporal data used for the running of hydrological models. Thus the database design needed to remain generic enough to allow for the inclusion of data from any hydrological models that may potentially be used. The geodatabase provides an existing platform on which to build a new data model, but is not suitable for the storage of extensive attribute data in an extensible manner. Likewise, Arc Hydro provides an established data model for the recording of data relating to hydrological features, however, is limited by inextensibility. Both the geodatabase and Arc Hydro lack the ability to store voluminous time series data in an efficient manner.

4.1 Design Criteria

With the geodatabase and Arc Hydro data model being used to meet the spatial and hydrological feature requirements, the focus of the project turned to the development of a new data model that would allow the integration of the aforementioned data models. Thus, a new data model specifically designed for the storage of hydrological attribute data was investigated. From the outset, it was decided that the new data model being developed would adhere to two fundamental criteria. The first was that the data model would have a generic design suitable for storage of data for use by any hydrological model, by making provision for the data types used in hydrological modelling, such as *single*, *text* and *boolean*. Secondly, the data model needed to be extensible, in order that the inclusion of new hydrological variables would not require changes to the structure of the data model. The new data model would also be required to be implemented within a suitable RDBMS supported by the geodatabase.

Apart from this, a number of more detailed stipulations had to be met. As mentioned in the Introduction, the new data model would need to:

- Store attributes for spatial features and non-spatial objects;
- Be able to include both input and output variables of the related hydrological models;
- Store a range of data types suitable for all hydrological models;
- Store units of measure for attributes where relevant; and
- Minimise the amount of data replication to maximise database storage efficiency.

The new data model was also required to be able to integrate with the ArcHydro data model as it enabled the modelling of connectivity of rivers and catchments, which would support functionality such as flow routing and network tracing. This would enable a user to perform tasks, such as finding connected or upstream catchments from a particular point on a river. It would also be necessary for there to be a geographic representation for the many hydrological features and their associated data that might be stored in an implemented database, such as the location of gauges, catchment outlets and abstraction points. Combined with this, would be the desired ability to be able to model the relationship between features (topology). This is illustrated by the idea that a Quaternary Catchment is part of an encompassing Tertiary, Secondary and thus Primary Catchment, as delineated in South Africa.

A final part of the design was the implementation of a time series storage method. Thus the outcome of Chapter 3 was important in the development of a time series component to the new data model that could store both regular and irregular interval time series data in an efficient manner, whilst remaining compatible with the rest of the data model.

The above design criteria were solved through the research into geographic data models, database models, the geodatabase, Arc Hydro, and time series, as described through the course of this dissertation. Several geographic data models were reviewed in Chapter 2, and provided insight into the advantages available through the use of the geodatabase. These included the advances in feature representation as illustrated by the geometric network, as well as the ability to store multiple geographic representations in a single database. Thus it was decided to use the geodatabase as the foundational geographic data model for this study, and thereby benefit from the object-relational database upon which it was built.

The Arc Hydro data model has the potential to be used in its capacity to model surface hydrology and hydrography, and provided a hydrological context to the development of the new data model. Arc Hydro would be included in order to model the many hydrological features and relationships that might be required, although the new data model under development would be designed to operate independently of Arc Hydro if considered necessary. Arc Hydro further provided a benefit in the form of a set of tools enabling the query and analysis of hydrologically related geographic data. Based on the evaluation of time series storage discussed in Chapter 3, it was decided to use the SPATSIM method of storing time series data. In addition, the SPATSIM data structure could be combined with

characteristics of the Arc Hydro time series structure, in order to improve efficiency and make the time series database more compatible with the Arc Hydro data model.

The goal of the design was thus the creation of a generic and extensible data model, within the confines of the geodatabase data model, which was to be suitable for the storage of data required for input into hydrological models. The extensible prerequisite for the design is restrictive when considering conventional geodatabase design, in which attribute data for each feature class is stored within a unique relational table. This is because the inclusion of additional spatial data in a geodatabase requires that a new feature class and table be created for the spatial data and its associated attribute data. Furthermore, the addition of any new attributes to an existing feature class requires the creation of a new field within the associated attribute table, thereby altering the data model structure, and thus violating the prerequisite for extensibility. Therefore the new data model would concentrate on the storage of attribute data, and in turn would result in the development of a new data model, *viz.* an attribute data model.

4.2 Working with the Geodatabase Data Model

The decision to work within the framework of ESRI's geodatabase data model was the result of numerous factors. First and foremost was the convenience of working with the geodatabase itself, since it was an existing geographic data model that had the support of an established developer of GIS software, and was also a widely used geographic data model. Thus, one of the advantages is that the geodatabase data model and its related components (such as the industry-specific data models like Arc Hydro, and the suite of GIS tools found in ArcGIS) are under continual revision to meet the demands of their users. Another important factor in the selection of the geodatabase was its foundation upon the object-relational database model. The object-oriented manner in which the geodatabase is structured also proved to be an influencing factor, in that it allows the new data model to take advantage of the key characteristics seen in this emerging approach, in particular, the ability to create relationships between tables that allows for the inheritance of qualities between tables.

4.3 The Arc Hydro Data Model

The Arc Hydro data model covers many aspects of hydrological data storage and modelling, and was selected as it was already an established hydrological data model for use with the geodatabase, and was fully integrated into ArcGIS. Since it was to be a significant component of this study, Arc Hydro's data model structure was also explored. As it was built upon the foundation of the geodatabase, the Arc Hydro data model used the object-relational data model. The data model also provided insight into the way in which the geodatabase data model managed geographic data, and thus provided clarity as to the necessary requirements of the new data model under development, in order for it to be compatible with both Arc Hydro and the geodatabase. Furthermore, ArcGIS provided a professional suite of GIS applications and thus, by conforming to its spatial standards as in the manner in which it records feature geometry, it would be possible to make use of its many tools.

To reiterate Section 2.4, the Arc Hydro data model is made up of *drainage*, *network*, *hydrography*, *channel* and *time series* components. Each of these were considered as to how the new data model would be able to interface with them, since it was preferable that minimal alteration of the Arc Hydro data model take place, as that would interfere with Arc Hydro's tools and the data model's functioning within ArcGIS. The primary element to take into consideration for the design of the new data model was Arc Hydro's *HydroCode*, as this attribute provided a potential link between the data model under development, since it distinguishes features from one another and provides a universal identifier for systems external to Arc Hydro. Illustrations of the Arc Hydro data model used in this study are provided in Figures 9.4 to 9.8 in the Appendix.

4.4 The Time Series Data Model

The storage of time series data was a critical part of the data model design, since the data model would be required to store large time series datasets often used in hydrological modelling. An important point to consider in the selection of a time series storage method was whether the chosen method would meet the design criteria, and whether it would be compatible with the remainder of the data model.

In the case of the dfs0 format reviewed in Section 3.5, there was a strict configuration governing its use. As a result, no alteration to the format could be made since it would interfere with the functioning of *Temporal Analyst* (the ArcGIS extension used for the importing of the dfs0 files). Furthermore, time series data stored in the dfs0 format would be in files external to the main database, since it was not compatible with the relational database model structure. Thus, if the dfs0 format were to be selected, some sort of link would need to be established between the data model under development and the dfs0 files.

As concluded in Chapter 3, the SPATSIM time series storage method was noted for its efficient storage of time series data with respect to the amount of disk space used, while the Arc Hydro time series storage method provided a logical storage structure from which the SPATSIM method could benefit. Both the SPATSIM and Arc Hydro time series storage methods have an added advantage of being directly compatible with the data model being developed, since they are both implemented within a relational database structure. A decision was thus made to employ the SPATSIM time series storage method in the remainder of this study, owing to its disk space efficiency.

Although the Arc Hydro method of time series was not chosen *per se*, it was decided that this method of time series storage provided the potential to improve on the SPATSIM method. This was possible through the use of Arc Hydro's *TSType* table, because of its efficiency in the recording of the attributes of each time series data. Since Arc Hydro provided a means by which to reduced duplicate time series attribute information, both the SPATSIM time series data model and Arc Hydro data model were combined. This was accomplished by including the key fields from the SPATSIM data model into a *Time Series* table within the Arc Hydro time series data model structure, as shown in Figure 4.1. The new time series data model used the structure of the Arc Hydro time series data model as detailed in Section 3.4, and took advantage of coded value domains. This enabled a simplification and reduction in the amount of disk space required for the storage of time series attribute information detailing the origin and format of the time series data. The integration of the SPATSTIM time series data model enabled the inclusion of BLOBs which greatly reduced the storage space used. This design provided an additional benefit in that the time series data model could now easily be integrated with the rest of the Arc Hydro data model, because of its similar configuration to the original time series data model of Arc Hydro.

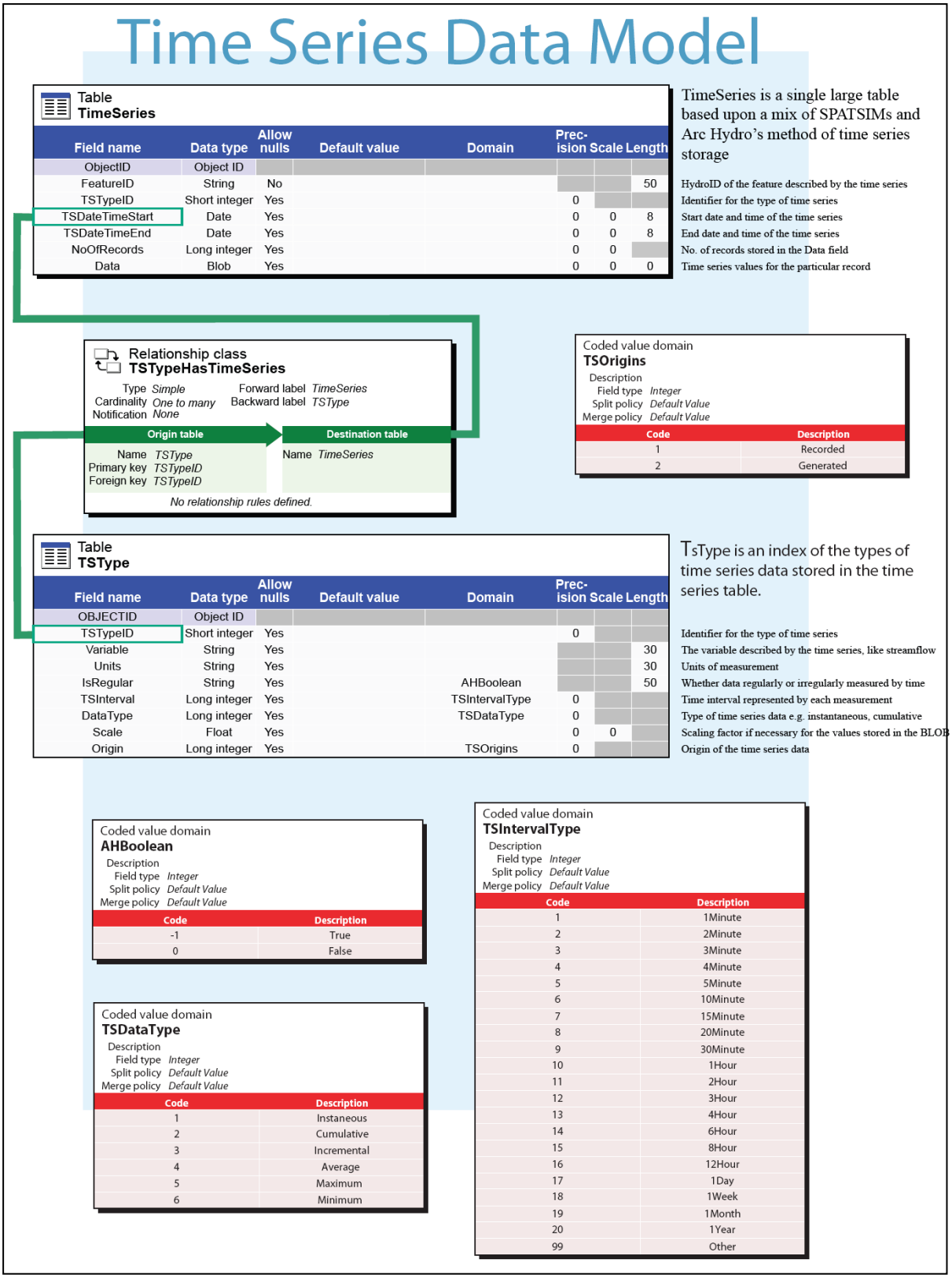


Figure 4.1 The combination of ArcHydro and SPATSIM methods of time series storage (modified after ESRI, 2003b)

4.5 Attribute Data Model Design

Concepts used in the design of the attribute data model were drawn from various sources. One of the primary influences on the design of the data model came about from an examination of the structure of the SPATSIM data model. Since the SPATSIM data model was built upon a relational database structure, and had been developed with the storage of spatial and time series data in mind, it offered many potential design elements that could be used in the creation of the new data model. One particular feature was the grouping of attribute data in SPATSIM into different attribute type tables, used for storage of data types commonly used in water resource modelling, as indicated in Section 3.4. The other influences on the design of the new attribute data model came from the evaluation of the time series data model and the Arc Hydro data model. The design of the data model was an iterative process, with a number of revisions being made before the final version was produced. The primary purpose for the attribute data model design was to create an extensible data model for the storage of hydrological data related to hydrological modelling. The data model therefore needed to be able to manage a variety of data types. One of the goals of the data model design was that there should be as little data replication as possible in order to increase database efficiency and reduce the disk space required.

In order to make provision for the various attribute data types of the hydrological data required for storage, a generic and extensible data model was designed. This data model makes use of eight different data type tables in order to store the various data types used by hydrological models, and provide a generic structure. These tables are, in turn, linked to two reference tables that provide a context to the data they contain. Each attribute to be stored in the data model is first recorded in the *MTBL_AttributeType* table, shown in Figure 4.2. This table contains a single record for each attribute, and the descriptive data related to it (including its description, units and default value). Thus in the case that additional attributes are to be added, one would only need to include them in the *MTBL_AttributeType* table. This table also serves to link each attribute with its particular data type table, through the use of the *DataTypeTable* field. This information is then linked with the various data type tables used in the attribute data model by using a table called *MTBL_AttributeData*. The *MTBL_AttributeData* table serves as a hub that unites the various other tables used in the data model. This connection is made possible by the creation of relationships between tables, which are linked through common fields. Thus, by using the *OBJECTID* field of the

MTBL_AttributeType table as the primary key, and the *AttributeType* field of *MTBL_AttributeData* table as the foreign key, it is possible to determine which of the data type tables correspond to the data held within the *MTBL_AttributeData* table. This is necessary, since without this knowledge it would not be possible to distinguish the location of the attribute data, as it would be unclear as to which data type table the relevant attribute data were held in. Thus in the case of the *OBJECTID* field of the *MTBL_AttributeType* table, a single record could relate to multiple records within the *AttributeType* field of the *MTBL_AttributeData* table. The actual recorded variable data is contained in a number of tables created for each of the possible data types, *viz.* Long Integer, Short Integer, Single, Double, Text, Date and Boolean data tables as shown in Figure 4.2 and 4.3. This information is then linked with the various data type tables used in the attribute data model by using the *MTBL_AttributeData* table.

Thus, additional relationships were created that employed the *OBJECTID* field of the particular data type table as the primary key, and the *RecordInTable* field of the *MTBL_AttributeData* table as the foreign key. The knowledge of which data type table and record in that table a particular hydrological attribute relates to, provides the means for the extraction of any selected attribute. Finally, the hydrological attribute in question can be linked to its original feature class (for example, *Quaternary* in Figure 4.2), through the *FeatureID* field found in the *MTBL_AttributeData* table. Given that the geometric data are to be stored within the Arc Hydro framework, the *FeatureID* can, in turn, be linked to the *HydroCode* of the associated feature class. Figures 4.2 and 4.3 provide clarity on the attribute data model's structure.

Time series data were stored in a hybrid format of SPATSIM and Arc Hydro, as mentioned previously, and illustrated in Figure 4.1. Its design, however, made it possible for the time series data model to function independently of the rest of the Arc Hydro data model, since the table could be linked to a separate feature class (such as *Quaternary)* via the *FeatureID* field of the *TimeSeries* table, and the Feature ID of an associated feature class such as the *MonitoringPoints* table in the *Hydrography* data model of Arc Hydro. At the same time, however, it could still be used independently of the Arc Hydro data model, and thus act as a ninth data type table in terms of the attribute data model, by using the time series data model's *ObjectID* as a primary key and the field *RecordInTable* within the *MTBL_AttributeData* table as the foreign key.

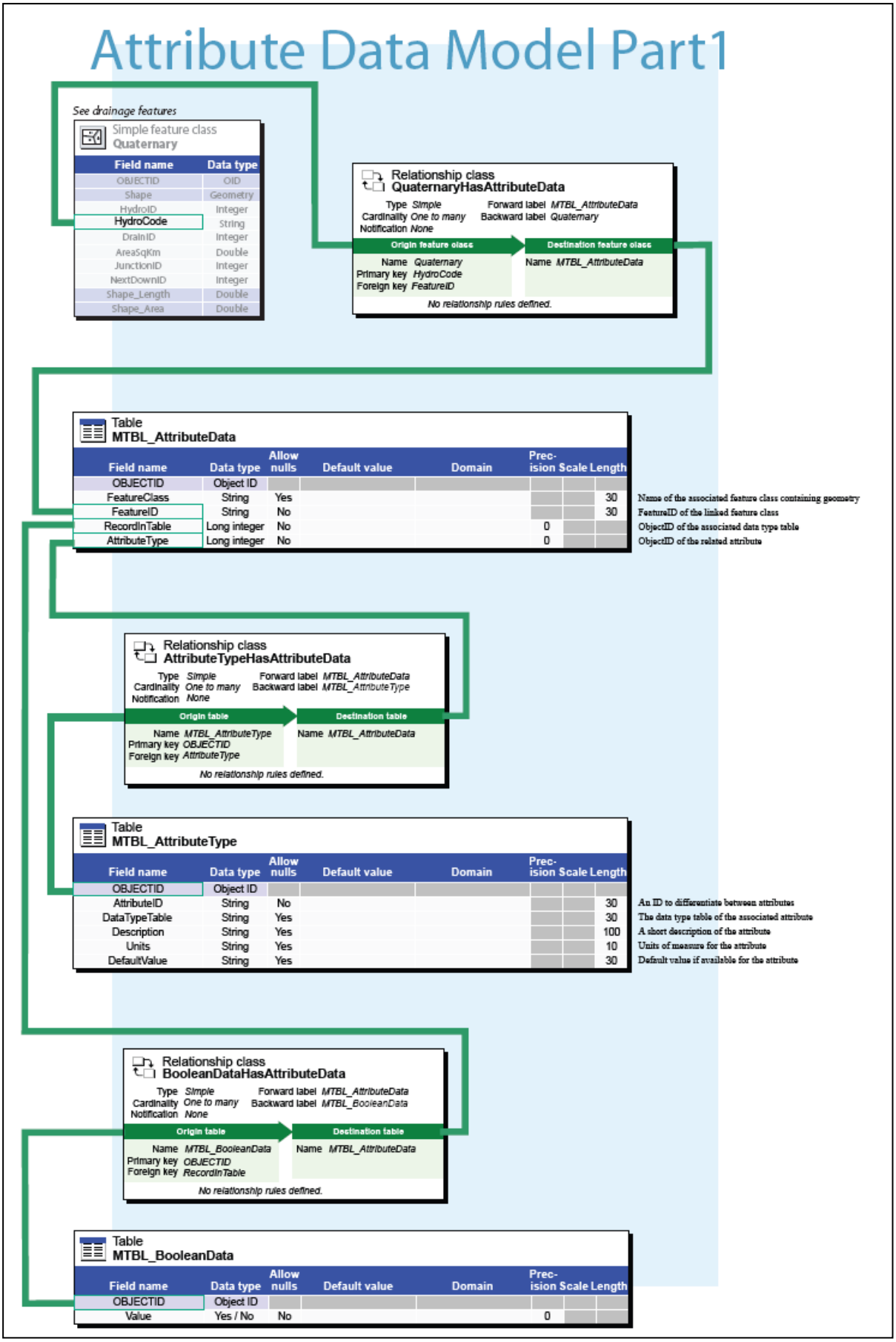


Figure 4.2 The structure of the attribute data model, Part 1

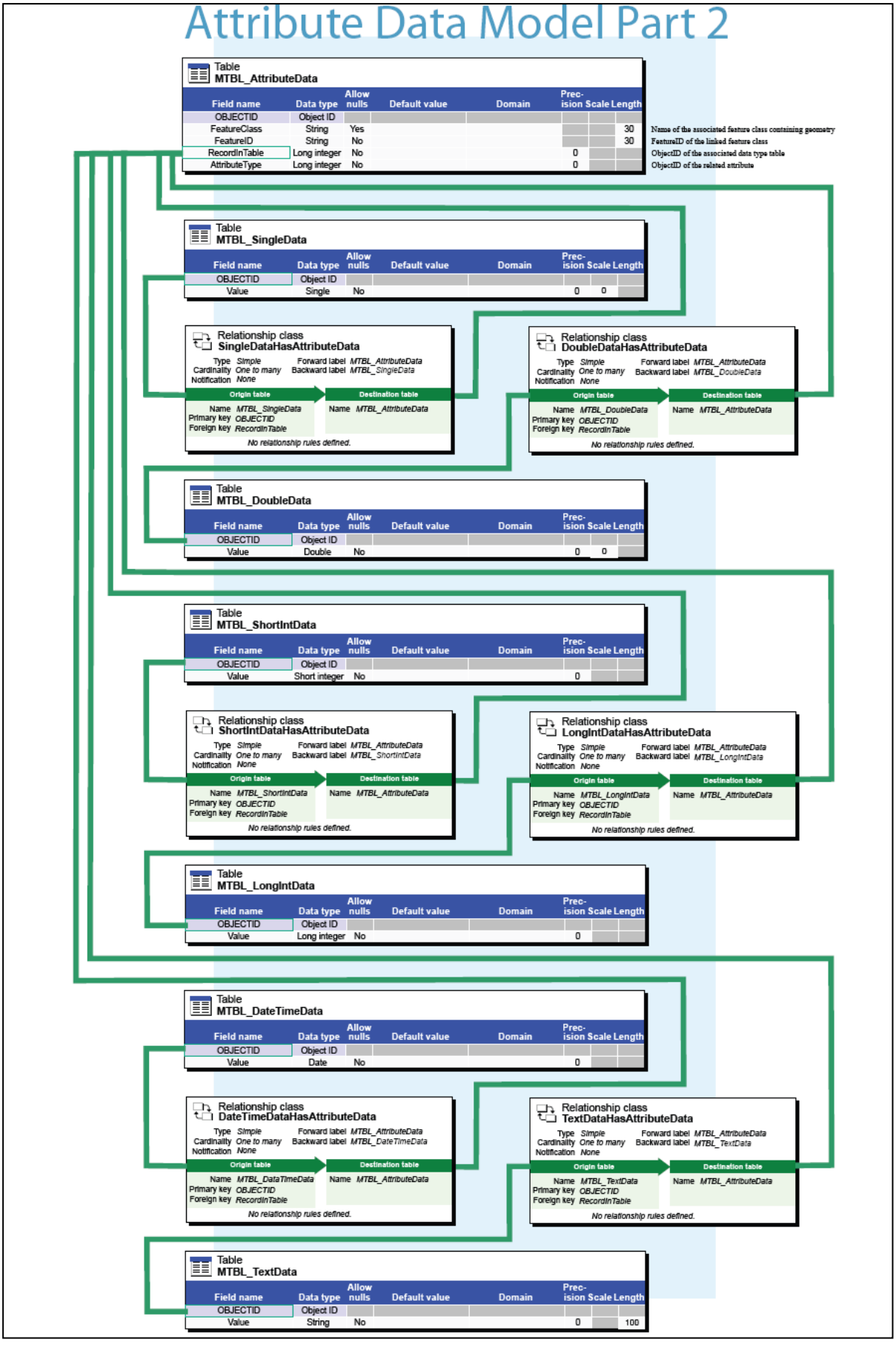


Figure 4.3 The structure of the attribute data model, Part 2

In the event that TIN and raster datasets were required for storage, they could be included alongside the data model as independent classes. Thus they would appear in their original format inside the geodatabase, and they would not interfere with the core model design and its functioning. This has not been illustrated in this study, since TINs and raster conventionally store only one attribute. Thus, storage of a single attribute within the attribute data model, related to a single geographic dataset (of TIN or raster), although possible, would likely be outweighed by the effort of conversion.

The design of the attribute data model creates a generic structure through the use of different data type tables, while the use of a *MTBL_AttributeType* table explicitly for the recording of attributes, allows for the addition of new hydrological variables to the data model without altering its structure, thereby making it extensible. The data model structure also provides for efficient storage of data as it minimises data replication by only recording attribute information once. A simplification of Figures 4.1, 4.2 and 4.3, is shown in Figure 4.4 below. The following chapter describes the implementation of the new data model design.

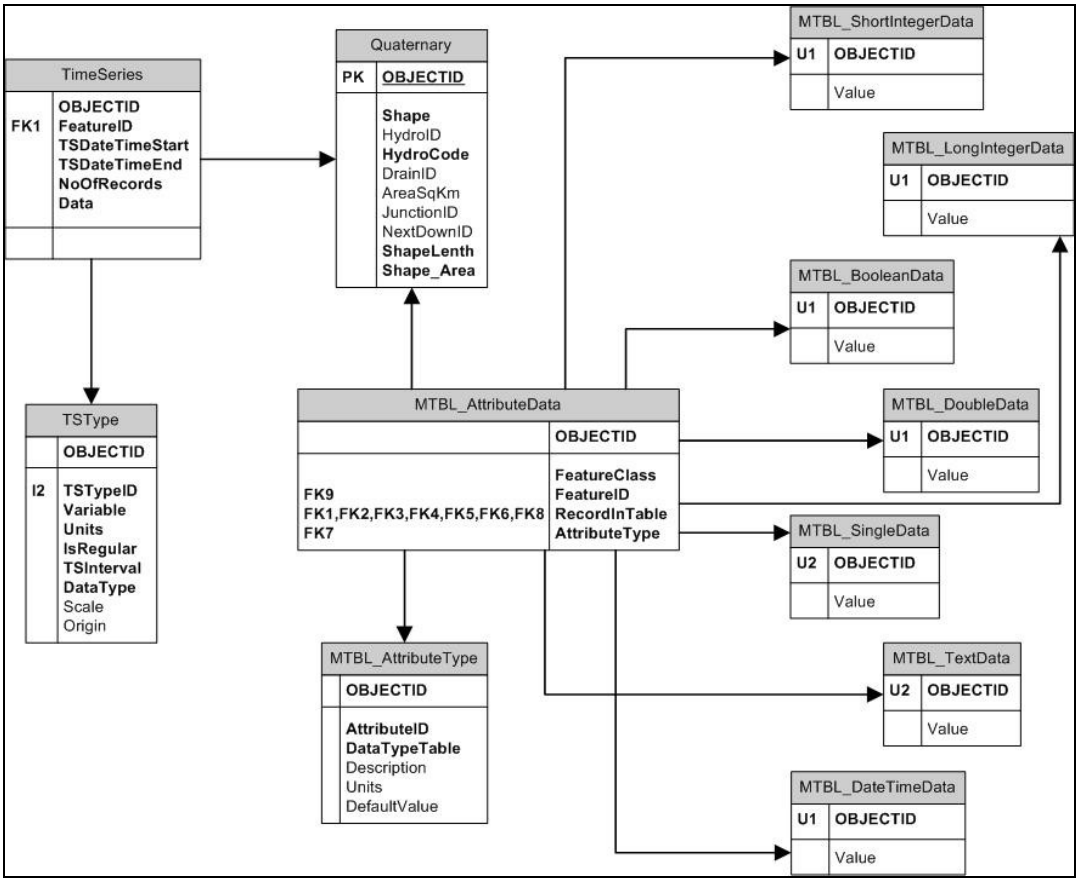


Figure 4.4. A database model (entity relational model) of the attribute data model

5. IMPLEMENTATION OF THE DATA MODEL

Once the design of the attribute data model had been completed, the next phase was to test the design by populating it with data, thereby creating a hydrological database. For this purpose, the South African Quaternary Catchments Database (QCDB) (Schulze *et al*., 2005), to which the ACRU agrohydrological modelling system (Schulze, 1995) is linked, was selected as a suitable hydrological dataset containing a variety of attribute data types, including time series data. The QCDB database has a simple structure from a spatial data perspective as it only contains a single Quaternary Catchment feature class. Microsoft Access was chosen as the RDBMS for the implementation of the data model, since it was readily available and supported the geodatabase, as explained in Section 4.2, and had been used in the original time series import process. The time series dataset for the QCDB, which had been investigated earlier in Chapter 3, would be used. Other than the geometric data provided by the QCDB, it would also be necessary to include additional geometric spatial data for use in completing the hydrological database with respect to feature topology and flow routing. This would be achieved through the population of Arc Hydro with the relevant feature datasets, including a river network and rainfall stations. Upon completion of the data loading, it would be necessary for tools to be developed in order to extract the necessary data from the database for the generation of ACRU menus. In addition, a tool for the creation of thematic maps, using data from the attribute data model, would be developed.

The goals for the implementation of the attribute data model are summarised in the following:

- Use the Arc Hydro data model for creation of feature classes for the geometric spatial data, and for the modelling of flow routing and feature topology;
- Include relevant spatial data such as Quaternary Catchment boundaries and rainfall station locations;
- Populate the data model with data from the QCDB (including time series);
- Be able to create thematic maps from data stored in the attribute data model, and thereby support additional spatial querying; and
- Include an application for the extraction of Quaternary Catchments data into a format suitable for the running of the ACRU model.

5.1 Quaternary Catchments Database

ACRU is an agrohydrological modelling system used in the modelling of the natural and anthropogenically influenced hydrological system (Schulze, 1995). The QCDB is a database used to create default ACRU menus for each of Southern Africa's 1 946 Quaternary Catchments. Use of the QCDB allows one to run the ACRU model and thereby generate various outputs, including streamflow peak discharge, groundwater recharge, sediment yield, irrigation water supply/demand and /or crop yields. An initial investigation of the QCDB was undertaken in order to determine its data storage requirements. This investigation revealed that a total of 644 ACRU variables would be required to be stored as different attributes. They consisted of numerical and textual values of varying precision and length, respectively. These attributes were then identified as to which attribute data type table they would be stored in. The 644 variables related to properties for each of South Africa's 1 946 Quaternary Catchments, including a hypothetical 1 947th catchment (the ocean) used for flow routing in the ACRU model. This made up a combined record set of 1 253 868 attribute values. Time series data for the QCDB had already been used in the investigation done in Chapter 3, and the complete time series dataset was prepared for import into the new database.

Since a single Quaternary Catchment is the representative container of spatial attribute data for all 644 attributes in the QCDB, a significant degree of spatial averaging is assumed, since the mean or modal values of spatially varying attributes such as soils characteristics are assigned to each Quaternary Catchment. Thus, in the case of soils information, the QCDB assumes average characteristics, despite soils typically not being of a uniform nature within any one Quaternary Catchment. This averaging is, however, necessary in order to reduce the complexity of modelling scenarios.

5.2 Design of an Implementation-Specific Data Model

For the purposes of this study only a portion of the Arc Hydro data model was required for use, since the QCDB only required spatial data in the form of Quaternary Catchments and monitoring point feature classes. Thus the channel component was excluded, while the *network*, *drainage*, *time series* and a portion of the *hydrography* component were included in revised formats. The *channel* component of the model was excluded since no channel data was present in the QCDB, In addition, hydrography data were limited to the monitoring

points represented by the rainfall stations, since this was the only hydrography data required with regards to the QCDB, and therefore the remainder of feature classes in the *hydrography* dataset could be omitted. In order to maximise detail, the figures showing the Arc Hydro data model in this section only portray those parts of the individual components of Arc Hydro that were used in this study. Figures 9.4 to 9.8 in the Appendix, provide a complete illustration of the Arc Hydro data model.

The hydrography feature dataset was simplified, as illustrated in Figure 5.1, and included a link between *MonitoringPoint* and the *TimeSeries* component, which had been altered as described previously in Section 4. The *Drainage* feature dataset was modified slightly in order to include the different levels of drainage features used in South Africa. This meant that the basin, watershed and catchment polygon feature classes of the original Arc Hydro data model were renamed to Primary, Secondary and Tertiary respectively. A forth drainage feature class was then added in the form of Quaternary, as shown in Figure 5.2, and thus would act as the representative feature class for the QCDB. Since temperature time series data in the QCDB are associated with each of the Quaternary Catchments, a relationship was created between the *Quaternary* and *TimeSeries* feature and object classes.

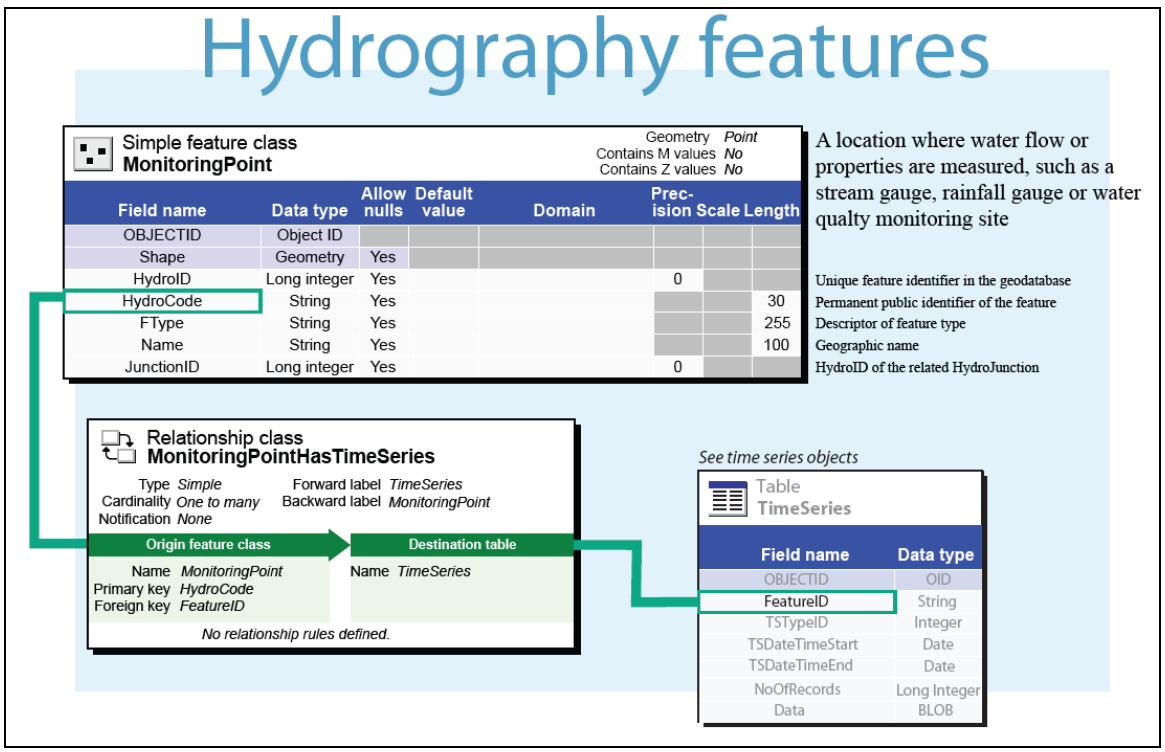


Figure 5.1 The portion of Arc Hydro's hydrography feature data model used for this study (modified after ESRI, 2003b)

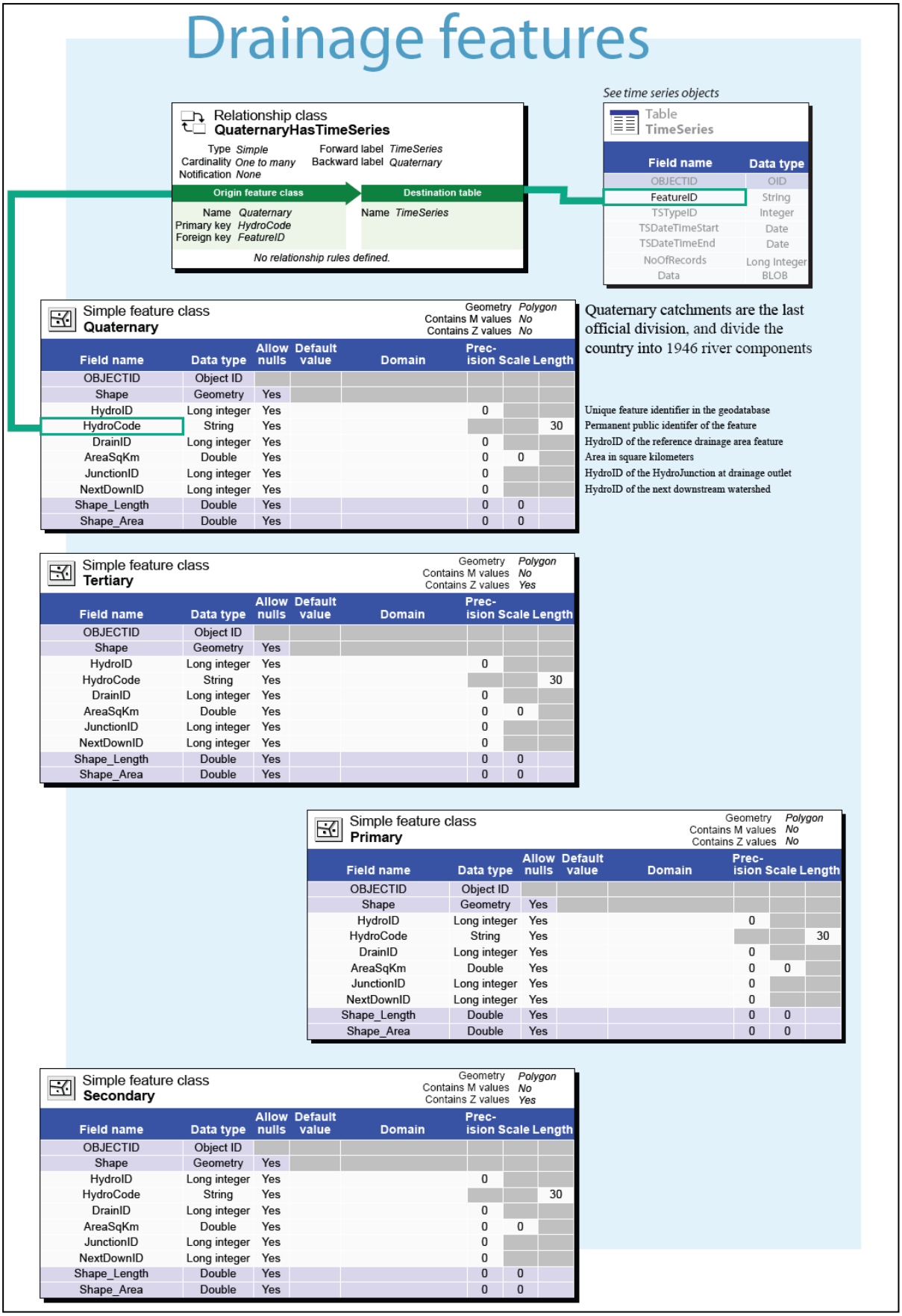


Figure 5.2 The customised drainage feature data model of Arc Hydro (modified after ESRI, 2003b)

The Arc Hydro data model assumes that related drainage feature classes are not necessarily spatially coincident with one another. However, since catchment divisions in South Africa are delineated by using their spatially coincident encompassing catchments, a number of topological rules could be added to increase the data integrity of the new database. These topological rules were defined in order to enforce data integrity, as shown in Figure 5.3, such that:

- Quaternary Catchments tessellate Tertiary catchments;
- Tertiary catchments tessellate Secondary catchments;
- Secondary catchments tessellate Primary catchments;
- Catchments do not overlap; and
- Catchments have no gaps.

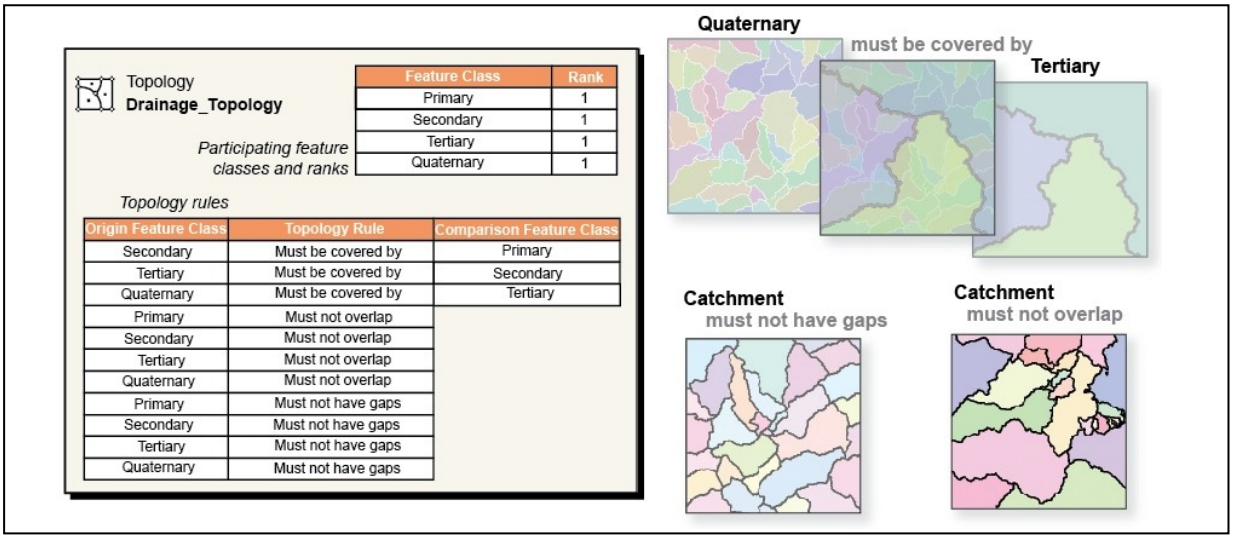


Figure 5.3 Topological rules of the drainage data model (modified after Arctur and Zeiler, 2004)

The *Network* dataset was the most utilised of the Arc Hydro model components in this study. The reason for this was that it provided the necessary structure for storing the geometric network data relating to Southern Africa's river system enabled the *Network* dataset to model the flow path between the Quaternary Catchments. Thus, although links between catchments were recorded in the QCDB as to which catchment was downstream of another, the *Network* feature dataset would make it possible to translate this into which rivers were upstream or downstream of each other, and to provide knowledge as to the segment of river reach found in each Quaternary Catchment. The *Network* dataset would also allow for the creation of a simplified river network (schematic), which could be used in modelling flow between Quaternary Catchments. Figure 5.4 shows the modified structure of the *Network* data model.

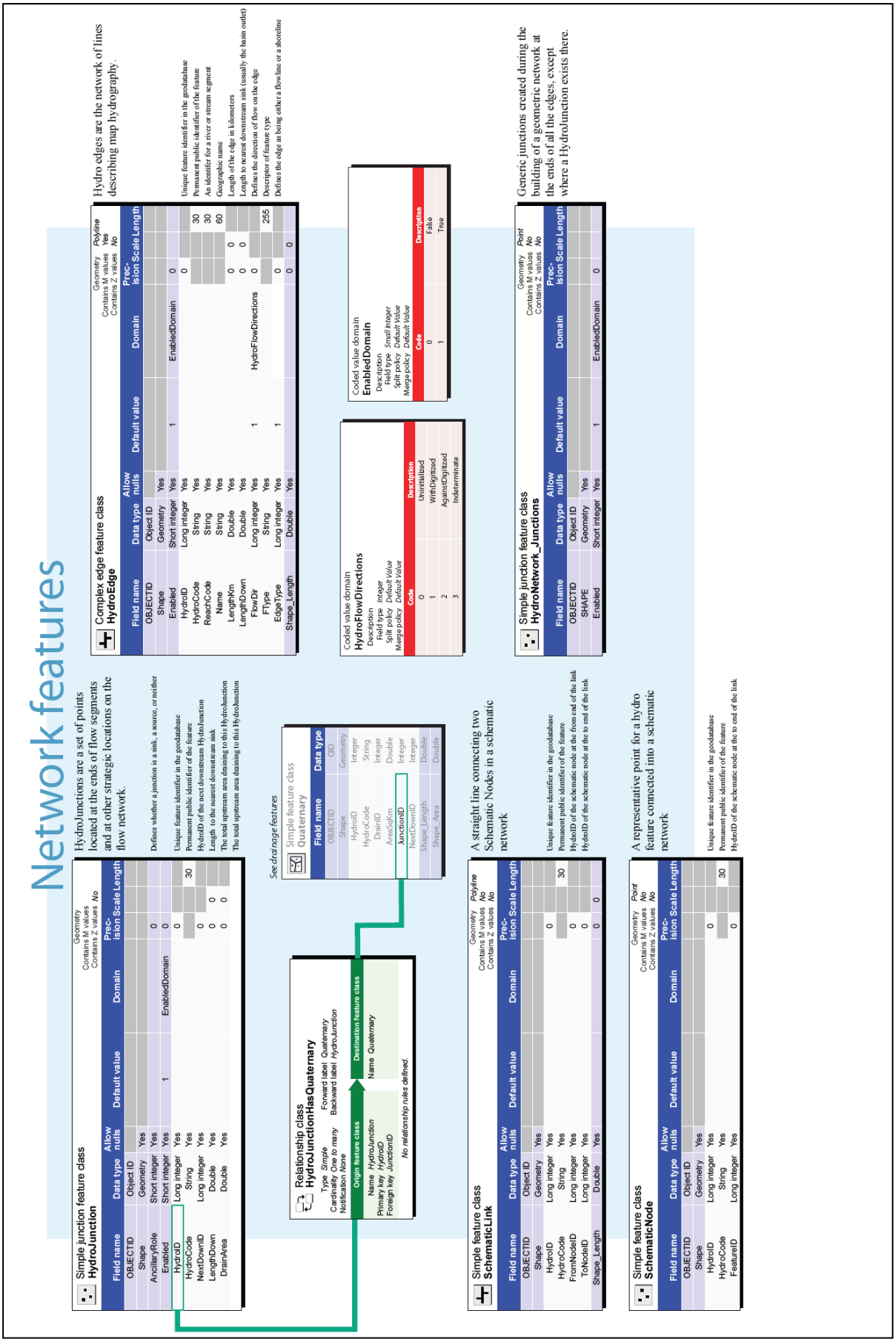


Figure 5.4 Adapted Arc Hydro Network features data model (modified after ESRI, 2003b)

5.3 Importing Geographic Data

A portion of the database population involved manipulation of related geographic datasets of the QCDB. A simple Quaternary Catchments feature class was first included that provided the polygon feature class for the QCDB, as well as the necessary drainage data for the Arc Hydro drainage feature data model represented in Figure 5.2. Thus, through the relationship between *FeatureID* in the *MTBL_AttributeData* table, and the *HydroCode* in the *Quaternary* feature class, it would be possible to perform spatial analysis on any of the attribute data stored in the attribute data model, as the two fields could be joined together, thereby providing the spatial link to the attribute data model. A point feature class was included that provided a spatial context for the rainfall time series data, and thus these rainfall monitoring stations were represented by *MonitoringPoints* in Arc Hydro's *Hydrography* dataset. Temperature time series data were associated to each one of the 1 946 Quaternary Catchments in the related feature class, through the *HydroCode* in the *Quaternary* feature class.

A river network was additionally required for potential flow routing and calculations involving hydraulic length of river segments. Thus a river network was used in the form of the Department of Water Affairs and Forestry's (DWAF) Quaternary River coverage. This coverage was first imported into a feature class, and was in turn simplified to include only those river reaches that linked up individual Quaternary Catchments, as is illustrated in Figure 5.5. This was done since only a basic flow routing at Quaternary Catchment level was required, and a simplification of the original river coverage thereby reduced complexity.

Once the simplified river network had been created, a geometric river network was generated in ArcGIS using the original river network as input. This geometric network was then manually edited in ArcGIS, and sinks in the network were defined, enabling flow direction to be calculated. With flow direction determined, it was now possible (using Arc Hydro's toolset) to perform tracing tasks on the geometric network, illustrated by finding those rivers connected to a particular reach, or the rivers (and by inference catchments) downstream of a point. This would prove useful in modelling, especially modelling involving point and non-point source pollution. Following the creation of a geometric network, a schematic network was created using ArcHydro's toolset. This was undertaken to further simplify river routing, and thus provide the simplest instance of catchment connectivity, as shown in Figure 5.6. A schematic network also enabled the representation of Quaternary Catchment flow routing, as

conceptualised within the QCDB. This completed the spatial data requirements for the implementation phase of the study.

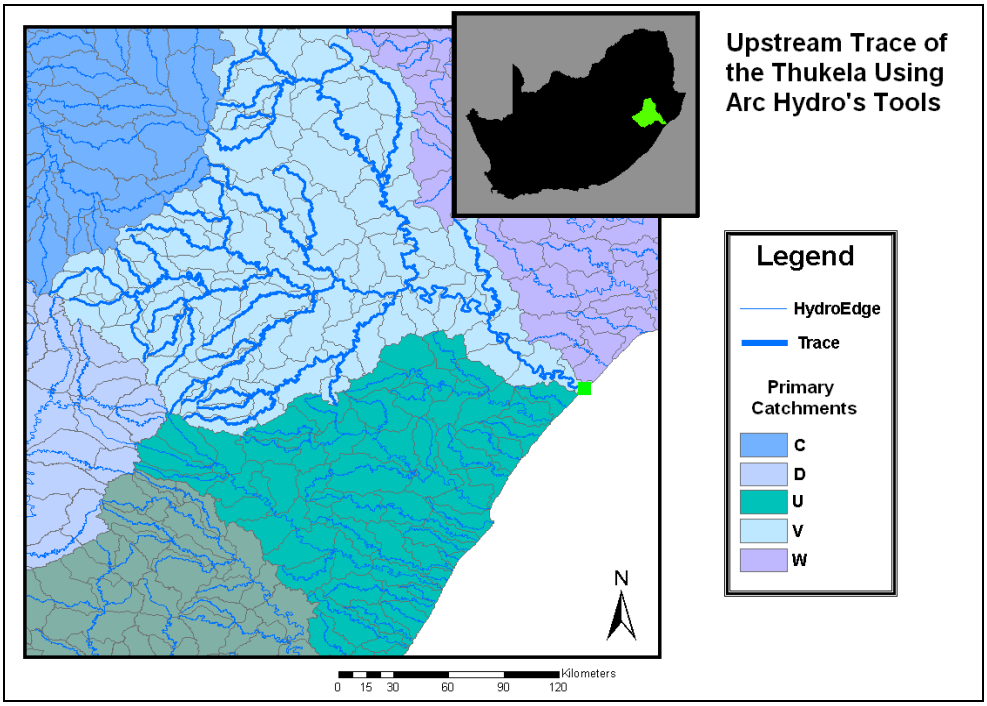


Figure 5.5 A portion of the Quaternary Catchments geometric river network showing an upstream trace

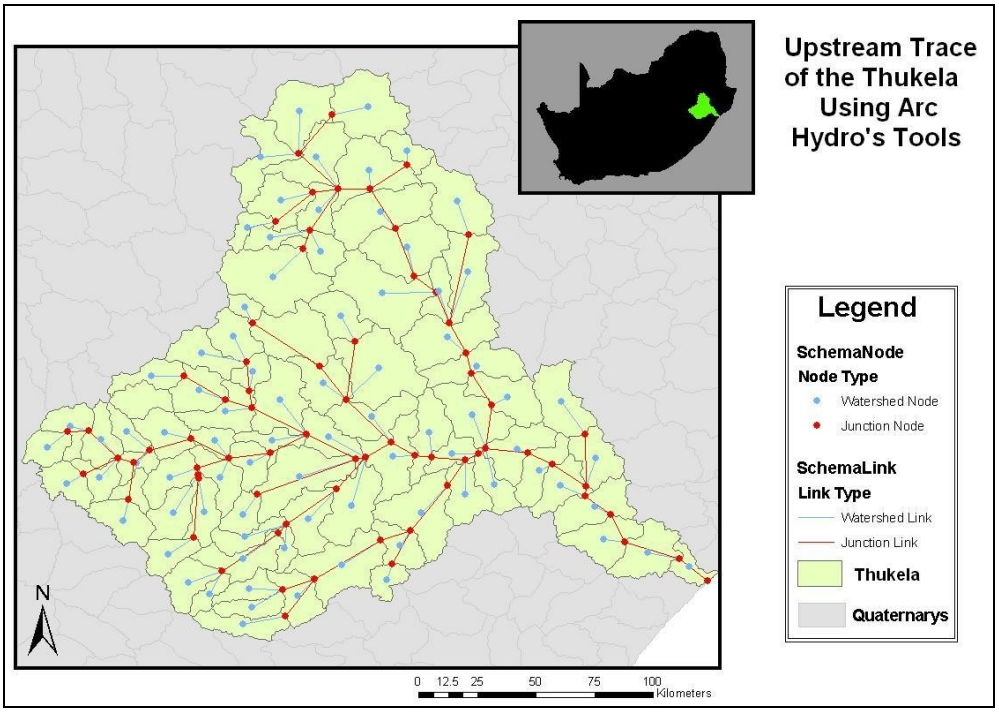


Figure 5.6 A schematic network of the Quaternary Catchments within the Thukela Primary Catchment

5.4 Importing Attribute Data

The importing of time series data into the attribute database was relatively simple, since the required import process had already been developed in Chapter 3. The time series database used was a combination of 1 244 unique rainfall station's records, as well as maximum and minimum temperature values for each of the 1 946 Quaternary Catchments, using the historical climate dataset described in Section 3.1 as the source of the time series data. Through the use of the previously developed import process, the entire climate database of 116 365 464 values was imported from its original ACRU composite format into a single *TimeSeries* table, as illustrated in Figure 4.1. Thus the time series of daily rainfall, minimum and maximum temperature for each station were recorded as individual records within the *TimeSeries* table, while their attribute information was recorded in the *TSType* table.

It must be noted at this point that the one difference between the previous import process in Chapter 3, and here, was that there was the requirement for the inclusion of time series data quality flag values. These data quality flags had been excluded from the original investigation into time series storage methods, since they were incompatible with some the dfs0 format. Data quality flag values were changed from string to numeric values in order to simplify importation, since numbers are easier to manage than text when considering SQL queries in a relational database. An additional advantage of the storage of numerical data quality flag values, was the reduction in disk space used. A lookup table matching the string flag values to the corresponding numeric data quality flag values was simultaneously created, in order to preserve the original string values for use during the subsequent extraction of time series data.

Static attribute data for each Quaternary Catchment in the QCDB were originally received in the form of three files with a comma delimited *CSV* file format. These files were imported into a temporary Microsoft Access table. An illustration of the format used is shown in Figures 4.2 and 4.3. Thereafter each of the 644 ACRU variables that were present were recorded in the *MTBL_AttributeType* table, along with the data type table its data would be stored in, and if available, its description, units and default values. The various ACRU variables data values were then carefully imported into their relevant data type tables, through the use of SQL. Thus blocks of 1 947 values for each of the Quaternary Catchments (including the ocean) were recorded in the attribute data model, for each ACRU variable. Finally, the *MTBL_AttributeData* table was populated with the relevant data that would link

up the other two tables of *MTBL_AttributeType* and the corresponding attribute data type table. With regard to the QCDB, all attribute data were related to the *Quaternary* feature class, and were recorded in the *FeatureClass* field of the *MTBL_AttributeData* table, while the *AttributeType* field recorded the ACRU variable in the *MTBL_AttributeType* table with which it corresponded. Finally, the *RecordInTable* field of the *MTBL_AttributeData* table established the necessary link between the ACRU variables and their location in the relevant attribute data type tables. Thus in order to extract a particular ACRU variable value, for a specific Quaternary Catchment, it would be necessary to know the *FeatureID* of the applicable Quaternary Catchment, the *DataTypeTable* it was stored in (and thus its *AttributeType*) and the *RecordInTable* value.

5.5 Data Analysis and Extraction Tools

After population of the database had been completed it was necessary to add data extraction capabilities, and thereby test whether the attribute data model functioned correctly. Two types of data extraction were needed, *viz.* for spatial visualisation, querying and analysis through ArcGIS, and then also for the creation of model input files for the ACRU model. The first of these two would constitute a generic extraction method, suitable for use with any hydrological attributes stored within the attribute database, while the second was expressly for the purpose of creating ACRU menus. Verification of the data model could then be undertaken by comparing outputs of the two methods of extraction with the original input data used in populating the attribute database, namely the QCDB.

As concluded in Section 5.4, the data contained in any one of the attribute data model tables is not meaningful when viewed as individual tables, and requires the use of the *MTBL_AttributeData*, *MTBL_AttributeType*, and associated data type tables, in order to make sense of it. The reason for this is that each of the aforementioned tables contain a portion of the data required for creating a complete dataset, for a specified attribute. The data type tables contain the recorded attribute values, while the *MTBL_AttributeType* table contains their corresponding attribute names. The *MTBL_AttributeData* table is additionally important, as it contains the *FeatureID* for any given attribute. Thus, by querying the database through the use of SQL, it is possible to generate a single table containing a list of desired attributes, and their associated *FeatureID*'s and values. This then allows for data in the attribute data model to be linked to a related feature class through the use of *FeatureID*.

This procedure is identical to how the database would be queried in ArcGIS in order to create any necessary attribute tables, and thereby join them to the relevant feature class. In this regard, the ArcObjects Developer Forum (ESRI, 2006) provided a SQL query tool for use in ArcGIS, which was modified in order to provide a simple way in which to query the attribute database and thereby generate the desired thematic maps. Figure 5.7 provides an example of the modified user interface where an attribute is specified from which the specified attribute table is generated. The user defines the workspace where the database is located, and then chooses the attribute to be extracted (an ACRU variable in this case). The option to *Always add OID* provides the choice as whether or not to include a unique ID in the creation of the new temporary table. The *Apply* button results in a query being generated and this populates an attribute data table in ArcGIS, with the specified *table name*. The table generated can subsequently be joined to the appropriate feature class through the *FeatureID* field, thus allowing for the creation of a thematic map using any associated hydrological attribute stored in the attribute data model.

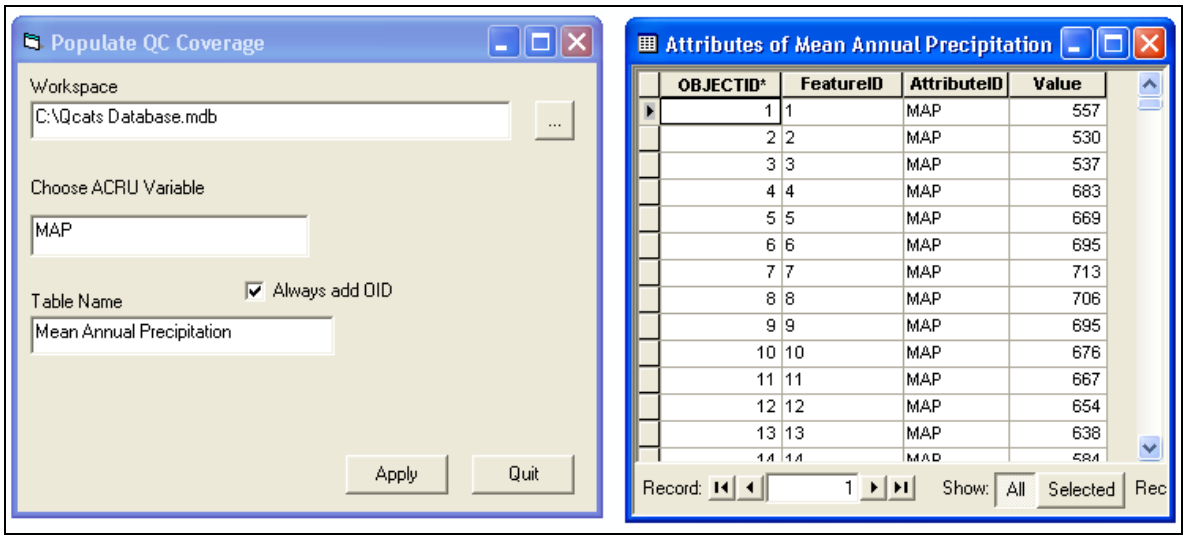


Figure 5.7 Creation of a mean annual precipitation table using the database query tool

The second method of extraction was required to generate suitable menus for the running of the ACRU model. This was achieved through a two-part process. The first part was the extraction of data from the attribute database, while the second part was the extraction of time series data from the time series component of the database. The reason for this split was due to the difference in storage methods and the required output formats of the two datasets. While the QCDB attribute data had been stored in various data type tables, the time series data had been placed in BLOBs that required Visual Basic code to access them, taking into

consideration the requirement to revert to the original data quality flag values as strings. In addition, the QCDB attribute data were required to be exported to their original CSV format, while time series needed to be placed in ACRU's composite file format. The application shown in Figure 5.8 was developed to enable the data required to run the ACRU model for a particular Quaternary Catchment's ID, or set of IDs, to be extracted. Then, with an option to include QCDB data, time series data, or both, the necessary data could be extracted for the creation of ACRU menus. This results in three *CSV* files being generated when *Include QCDB Data* is selected, and the related number of time series composite files if *Include Time Series Data* is selected, thus placing the data back into its original format. These files were subsequently compared with the original input files in order to determine whether the data model had stored and processed them correctly, and they were found to be identical. With the ability to export data from the attribute database into suitable input files for the running of the ACRU model, the goals outlined in the implementation of the data model at the beginning of this chapter were completed, thereby concluding the study. An example of the visual basic code used in the exportation of data from the database, including an integrated SQL clause, is shown in Figure 9.8 of the Appendix.

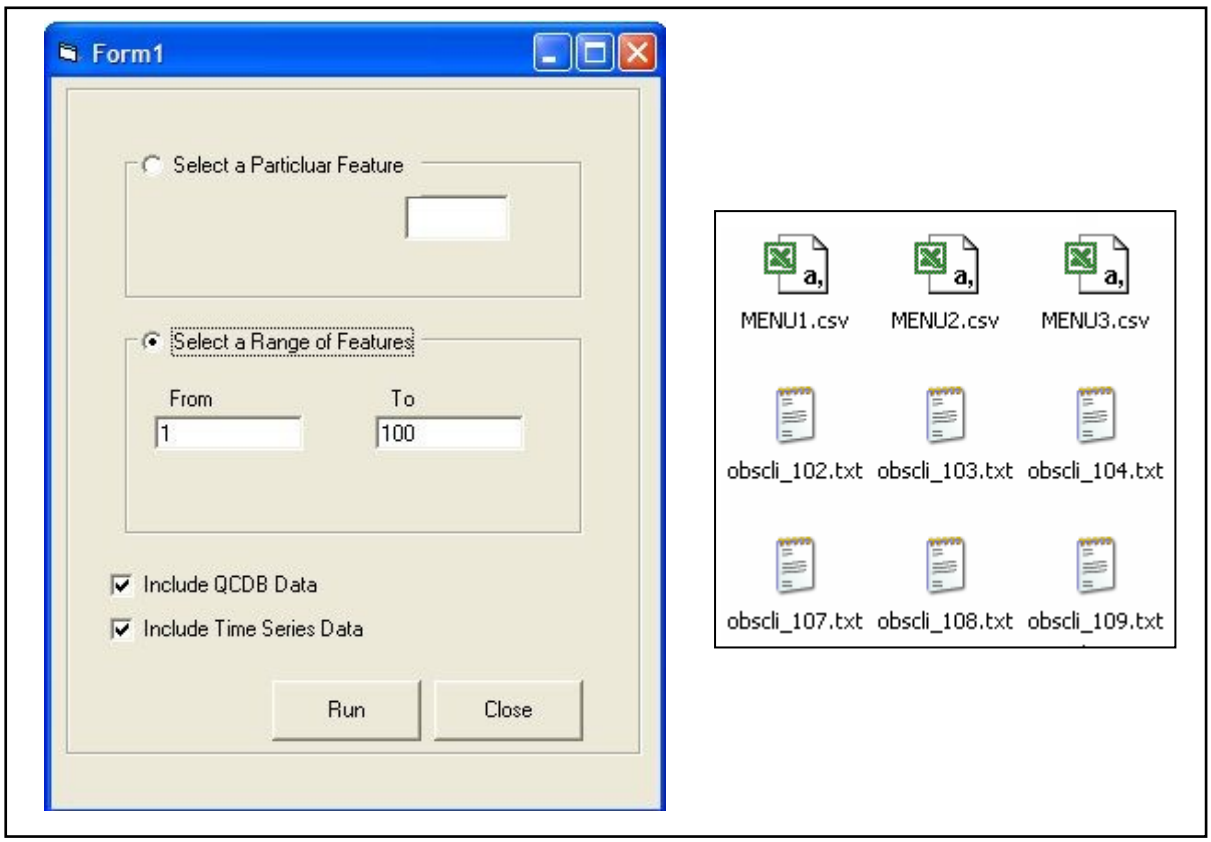


Figure 5.8 The user interface for the extraction of the QCDB data into its required formats

6. DISCUSSION AND CONCLUSION

The initial review on the history of geographic data models provided perspective with respect to what the geodatabase could offer in comparison, while the database data models, and the structure of the geodatabase, enabled the evolution of ideas as to how the new data model could be designed and implemented. Earlier geographic data models to the geodatabase have been limited in their ability to represent multiple geographic representations models concurrently and also stored their spatial and attribute data separately. Advances in databases, have enabled the development of new technologies to aid in the storage of geographic data, evident in the creation of the object-relational data model. Previous limitations have been overcome with the geodatabase since it allows for the storage of multiple spatial representations and their spatial and attribute data within a single database, while adding functionality through the combination of behaviour with features. The geodatabase is, however, unsuitable for the storage of attribute data in an extensible manner, and thus an alternative was required.

The investigation into time series storage methods in Chapter 3 resulted in the identification of a suitable method to store time series data with respect to the evaluation criteria. This included the storage of data in an efficient manner, and enabled both regular and irregular interval time series to be held within a single repository. The selection of the SPATSIM method of time series storage enabled the development of a time series database with a logical design and minimal use of disk space. Since Arc Hydro was to be a significant part of this study, the SPATSIM method was incorporated into the time series component of this data model, as it allowed for easier integration of time series with the Arc Hydro and attribute data models, and could take advantage of Arc Hydro's logical design.

The utilisation of the Arc Hydro data model in the modelling of the geographical context of features and objects was particularly suitable owing to the data model's focus on hydrology. Thus spatial features such as gauges and catchments could be represented in the GIS. The use of Arc Hydro enabled the modelling of connectivity between rivers and catchments, and could therefore be used in flow routing and network tracing. The design of the new data model was thus influenced by SPATSIM and Arc Hydro. By employing Arc Hydro in the modelling of spatial data, advances in the design of a new data model could continue, and ultimately

resulted in the formation of a new attribute data model. This new attribute data model enabled the storage of attributes for spatial features and improved on previous designs as the data model was extensible, such that the addition of any new model variables did not require alteration to its structure. Furthermore, database efficiency was maximised by minimising data replication. The attribute data model extended the Arc Hydro data model enabling the storage of attribute and time series data in an extensible and efficient way.

Upon the completion of the design component for the project it was then necessary to validate the attribute data model, through its implementation, in the creation of a hydrological database. By selecting the QCDB, which contained significant amounts of attribute data associated with the ACRU model, it was possible to create a complete hydrological database using numerous data types. In addition, the full historical climate dataset of the QCDB was used as a source of time series data, and was tested by implementing it within the modified time series storage data model based upon SPATSIM and Arc Hydro. Thus both the time series data model and the attribute data model were populated with relevant data. In order to test the spatial link between the attribute data model and Arc Hydro, the related spatial datasets were added. The Arc Hydro data model was thus populated, although in a revised form, since only a portion of the Arc Hydro data model was required for the purposes of this study.

Lastly, it was necessary to develop tools to enable data in the time series and attribute data models, to be viewed and exported. A generic export tool was developed for the creation of thematic maps, using data from the attribute data model and joining it with the associated spatial feature class. This enabled the functionality offered by ArcGIS to be applied, and thus common GIS operations such as the spatial querying of geographic data could be used. A study-specific export tool was further added for the purpose of ACRU menu generation. The tool made use of both the time series and attribute databases, and produced the files specified for use by the ACRU model.

With reference to the original design criteria for the project, the goal of designing a data model for the storage of hydrological data has been met. The geodatabase and Arc Hydro data models were extended by the attribute data model and the time series data model, enabling the efficient storage of large attribute datasets in a generic and extensible manner. Overall, the data models used created a spatial context for the storage of geographically related hydrological data within a single database, for use in future hydrological modelling.

7. RECOMMENDATIONS FOR FUTURE RESEARCH

Given the evolutionary nature of GIS and databases, there is always potential for more efficient storage of data. Thus in terms of this study, the geodatabase was the container for all the data stored. However, with the development of truly object-oriented GIS databases, there would be the potential for a more logical design related to the attribute data model, since external GIS object and feature classes could more closely resemble their internal database counterparts. This potential future upgrade, as well as the current instance of the attribute data model, depends on the proprietary software of ArcGIS and would benefit from a tailored GIS that could operate independent of software licences.

As it exists, the attribute data model developed is unable to take full advantage of the validation rules present in a conventional geodatabase. Therefore a recommendation for future research would be for the inclusion of validation rules to the importation of data into the attribute data model, based on either data ranges, or other data entry criteria. This brings to the fore one of the main difficulties in applying the attribute data model, namely the importation of data. This is as a result of no particular import process being developed, since hydrological data models differ in their data storage formats and requirements. Thus the population of the data model requires knowledge of its structure. A subsequent improvement would therefore be the development of a generic import procedure supporting the data formats common used by hydrological models.

The Arc Hydro data model used in this project concentrates on surface water flows, and for the most part excludes the groundwater component of the hydrological system. An upgrade to the Arc Hydro data model has now been developed which extends the Arc Hydro data model by integrating groundwater modelling (Strassberg, 2005). The inclusion of this groundwater data model component would create a complete data model with regards to hydrology and its representation through GIS.

8. REFERENCES

AGI, 2006. GIS Dictionary. [Internet]. Association for Geographic Information. Available from: http://www.geo.ed.ac.uk/agidexe/term?727 [Accessed: August, 2006].

Arctur, D and Zeiler, M. 2004. *Designing the geodatabase: Case studies in GIS data modelling*. Environmental Systems Research Institute, Redlands, USA.

Batty, P. 1991. Why Use a Single Database Management System for GIS. In: ed. Parker, DH., *International GIS Sourcebook.* GIS World, Inc. Fort Collins, USA.

Bech, T. 2005. Personal Communication. Danish Hydrological Institute, Hørsholm, Denmark, November, 2005.

Briggs, R. 2004. GIS Management and Implementation. [Internet]. University of Texas. Available from: http://www.utdallas.edu/~briggs/poec6383/dbconcept.ppt [Accessed: September, 2004].

Chou, Y. 1997. *Exploring Spatial Analysis in Geographic Information Systems*. OnWord Press, Santa Fe, USA.

Coppock, JT and Rhind, DW. 1991. The History of GIS. In: eds. Goodchild, MF, Maguire, D J, and Rhind, DW, *Geographical Information Systems - Principles and Applications,* Chapter 7. Longman Scientific & Technical, New York, USA.

Clark, DJ. 2005. Personal communication, School of Bioresources and Environmental Engineering, at the University of KwaZulu-Natal, Pietermaritzburg, RSA, June 2005.

Dangermond, J and Schutzberg, A. 1998. Editorial. *Journal of Computing in Civil Engineering* 12(3):121-122.

Databasedev.co.uk, 2006. Working with Database Fields - Database Solutions for Microsoft Access. [Internet]. Databasedev.co.uk – Database solutions and downloads for Microsoft Access. Available from: http://www.databasedev.co.uk/fields_datatypes.html [Accessed: May 2006].

DeMers, MN. 2000. *Fundamentals of Geographic Information Systems*. Wiley, New York, USA.

DHI. 2006. MIKE Flood. [Internet]. Danish Hydrological Institute, Hørsholm, Denmark. Available from: http://www.dhisoftware.com/mikeflood/index.htm [Accessed: November 2006]

ESRI, 2002. Working with the Geodatabase: Powerful Multiuser Editing and Sophisticated Data Integrity. [Internet]. Environmental Systems Research Institute, Redlands, USA. Available from: www.esri.com [Accessed: August, 2004].

ESRI, 2003a. Spatial Data Standards and GIS Interoperability. [Internet]. Environmental Systems Research Institute, Redlands, USA. Available from: www.esri.com [Accessed: July, 2006].

ESRI, 2003b. The ArcGIS Hydro Data Model. [Internet]. Environmental Systems Research Institute, Redlands, USA. Available from: www.esri.com [Accessed: July, 2006].

ESRI, 2004. RDBMS Concepts of a Geodatabase. [Internet]. Environmental Systems Research Institute, Redlands, USA. Available from: www.esri.com [Accessed: July, 2006].

ESRI, 2006. RDBMS Concepts of a Geodatabase. [Internet]. Environmental Systems Research Institute, Redlands, USA. Available from: http://edndoc.esri.com /arcobjects/8.3/ [Accessed: July, 2006].

Evans, S and Millett, N. 2002. Working with the Geodatabase - Scalable GIS Solutions for the Hydrographic Community. [Internet]. ESRI. Available from: www.thsoa.org [Accessed: August, 2004].

Hallowes, LA, Schulze, RE, Horan, MJC and Pike, A. 2004. South African National Quaternary Catchments Database: Refinements to, and links with, the *ACRU* model as a framework for installed hydrological modelling systems. In: eds. Schulze, RE and Pike, A, *Development and Evaluation of an Installed Hydrological Modelling System*, Chapter 6, 93-120. WRC Report 1155/1/04, Water Research Commission, Pretoria, RSA.

Harmon, JE and Anderson, SJ. 2003. *The Design and Implementation of Geographic Information Systems*. John Wiley & Sons, Hoboken, USA.

HDS, 2001. GIS Data Formats and their Conversion. [Internet]. Harvard Design School. Available from: http://www.clarku.edu/faculty/marcano/geo206/gis_data_formats.htm [Accessed: September, 2004].

Healey, RG. 1991. Database Management Systems. In: ed. Maguire, DJ., Goodchild, MF., and Rhind, DW., *Geographical Information Systems : Principles and Applications Volume 1*, 6, 251-267. Longman, Harlow, UK.

Hipel, KW and McLeod, AI. 1994. *Time Series Modelling of Water Resources and Environmental Systems*. Elsevier, Amsterdam, Netherlands.

Hughes, D and Forsyth, D. 2002. SPATSIM – Spatial and Time Series Modelling Software. Institute of Water Research. Rhodes University, Grahamstown, RSA.

IWR. 2004. SPATSIM - Spatial and Time Series Information Modelling Software. [Internet]. Institute for Water Research, Rhodes University. Available from: http://www.ru.ac.za/institutes/iwr/software/spatsim.html [Accessed: November 2005].

Khoshafian, S. 1993. *Object-Oriented Databases*. John Wiley & Sons, New York, USA.

Leung, Y. 1997. *Intelligent Spatial Decision Support Systems*. Springer-Verlag, Berlin, Germany.

Maguire, DJ. 1991. An Overview and Definition of GIS. In: ed. Goodchild, MF, Maguire, DJ, and Rhind, DW. *Geographical Information Systems - Principles and Applications,* ch. Longman Scientific & Technical, New York, USA.

Maidment, D. 2002. *Arc Hydro - GIS for Water Resources*. Environmental Systems Research Institute, Redlands, USA.

Microsoft, 2007. FAT16 vs. FAT32. [Internet]. Microsoft. Available from: . http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/core/fncc_fil_blpd.mspx?mfr=true [Accessed: December 2007].

Mitášová, I and Višòovcová, MJ. 2002. *Approaches to Data Modelling in GIS*. Laboratory of Geoinformatics Department of Theoretical Geodesy at Slovak Technical University, ArtInAppleS Ltd, Bratislava, Slovakia.

Nabar, P and Patel, D. 2003. GIS Database Options. [Internet]. University of Texas at Dallas, Dallas, US. Available from: http://charlotte.utdallas.edu/mgis/ClassFiles/gisc6383/techassess_2004/GIS- DATABASE-OPTIONS.ppt [Accessed: October, 2006].

NCES. 2006. Projections of Education Statistics to 2013. [Internet]. National Center for Education Statistics. Available from: http://nces.ed.gov/programs/projections/appendix_D.asp [Accessed: January 2006].

O' Neil, P. 1994. *Database Principles Programming Performance*. Morgan Kaufmann Publishers, San Francisco, USA.

Raza, A. 2001. Object-Oriented Temporal GIS for Urban Applications. Published PhD Dissertation, University of Twente, Enschede, Netherlands.

Rigaux, P Scholl, M, and Voisard, A. 2002. *Spatial Databases: With Application to GIS*. Morgan Kaufmann, San Francisco, USA.

Rob, P and Coronel, C. 1997. *Database Systems: Design, Implementation and Management*. Course Technology, Cambridge, UK.

Schulze, RE. 1995. *Hydrology and Agrohydrology: A Text to Accompany the ACRU Agrohydrological Modelling System.* Report Number TT69/95, Water Research Commission. Pretoria, RSA.

Schulze, RE, Warburton, M, Lumsden, TG and Horan, MJC. 2005. The Southern African Quaternary Catchments Database: Refinements to, and Links with, the *ACRU* System as a Framework for Modelling Impacts of Climate Change on Water Resources. In: ed. Schulze, RE, *Climate Change and Water Resources in Southern Africa: Studies on Scenarios, Impacts, Vulnerabilities and Adaptation*, Chapter 8, 111-139. WRC Report 1430/1/05, Water Research Commission, Pretoria, RSA.

Schulze, RE, Hallowes, LA, Horan, MJC, Lumsden, TG, Pike, A., Thornton-Dibb, S and Warburton, ML. 2006. South African Quaternary Catchments Database. In: ed. Schulze, RE, *South African Atlas of Climatology and Agrohydrology,* Section 2.3. WRC Report 1489/1/06, Water Research Commission, Pretoria, RSA.

Singh, VP and Fiorentino, M. 1996. *Geographical Information Systems in Hydrology*. Kulwer Academic Publishers, Dordrecht, Netherlands.

Smithers, JC, Dent, MC, Lynch, SD and Schulze, RE. 1995. Preparation of Daily Climatic Input Files. In: ed. Schulze, RE, *Hydrology and Agrohydrology: A Text to Accompany the ACRU 3.00 Agrohydrological Modelling System*, AM4-1 to AM4-2. WRC Report TT69/95. Water Research Commission, Pretoria, RSA.

Strassberg, G. 2005. A Geographic Data Model for Groundwater Systems. Unpublished PhD thesis, University of Texas at Austin, Austin, USA.

Twumasi, BO. 2002. Modelling Spatial Object Behaviours in Object-Relational Geodatabase. Unpublished MSc Dissertation, International Institute for Geo-information Science and Earth Observation, Enschede, Netherlands.

Wachowicz, M. 1999. *Object-Oriented Design for Temporal GIS*. Taylor & Francis, London, UK.

Worboys, M. 1995. *GIS: A Computing Perspective*. Taylor & Francis, London, UK.

Zeiler, M. 1999. *Modeling Our World - The ESRI Guide to Geodatabase Design*. Environmental Systems Research Institute, Redlands, USA.

Zeiler, M. 2001. *Exploring Arc Objects – Volume 1: Applications and Cartography*. Environmental Systems Research Institute, Redlands, USA.

9. APPENDIX

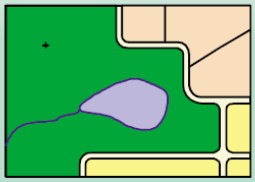
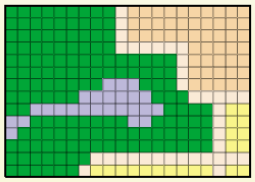
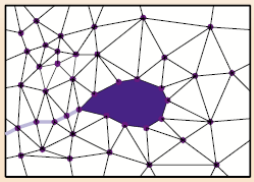
	Vector data representation	Raster data representation	Triangulated data representation
			
Focus of model	Vector data is focused on modeling discrete features with precise shapes and boundaries.	Raster data is focused on modeling continuous phenomena and images of the earth.	Triangulated data is focused on an efficient representation of a surface that can represent elevation or other quality, such as concentration.
Sources of data	Compiled from aerial photography Collected from GPS receivers Digitized from map manuscripts Sketched on top of raster display Vectorized from raster data Contours from triangulation Reduced from survey field data Imported from CAD drawings	Photographed from an airplane Imaged from a satellite Converted from a triangulation Rasterized from vector data Scanned blueprints, photographs	Compiled from aerial photography Collected from GPS receivers Imported points with elevations Converted from vector contours
Spatial storage	Points stored as x,y coordinates. Lines stored as paths of connected x,y coordinates. Polygons stored as closed paths.	From a coordinate in the lower-left corner of the raster and cell height and width, each cell is located by its row and column position.	Each node in a triangle face has an x,y coordinate value.
Feature representation	Points represent small features. Lines represent features with a length but small width. Polygons represent features that span an area.	Point features are represented by a single cell. Line features are represented by a series of adjacent cells with common value. Polygon features are represented by a region of cells with common value.	Point z values determine the shape of a surface. Breaklines define changes in the surface such as ridges or streams. Areas of exclusion define polygons with the same elevation.
Topological associations	Line topology keeps track of which lines are connected to a node. Polygon topology keeps track of which polygons are to the right and left sides of a line.	Neighboring cells can be quickly located by incrementing and decrementing row and column values.	Each triangle is associated with its neighboring triangles.
Geographic analysis	Topological map overlay Buffer generation and proximity Polygon dissolve and overlay Spatial and logical query Address geocoding Network analysis	Spatial coincidence Proximity Surface analysis Dispersion Least-cost path	Elevation, slope, aspect calculations Contour derivation from surface Volume calculations Vertical profiles on alignments Viewshed analysis
Cartographic output	Vector data is best for drawing the precise shape and position of features. It is not well suited for continuous phenomena or features with indistinct boundaries.	Raster data is best for presenting images and continuous features with gradually varying attributes. It is not generally well suited for drawing point and line features.	Triangulated data is best for rich presentation of surfaces. This data can be viewed by using color to show elevation, slope, or aspect or in a three-dimensional perspective.

Figure 9.1 Comparison of characteristics for the three spatial representations of geographic data (Zeiler, 1999)

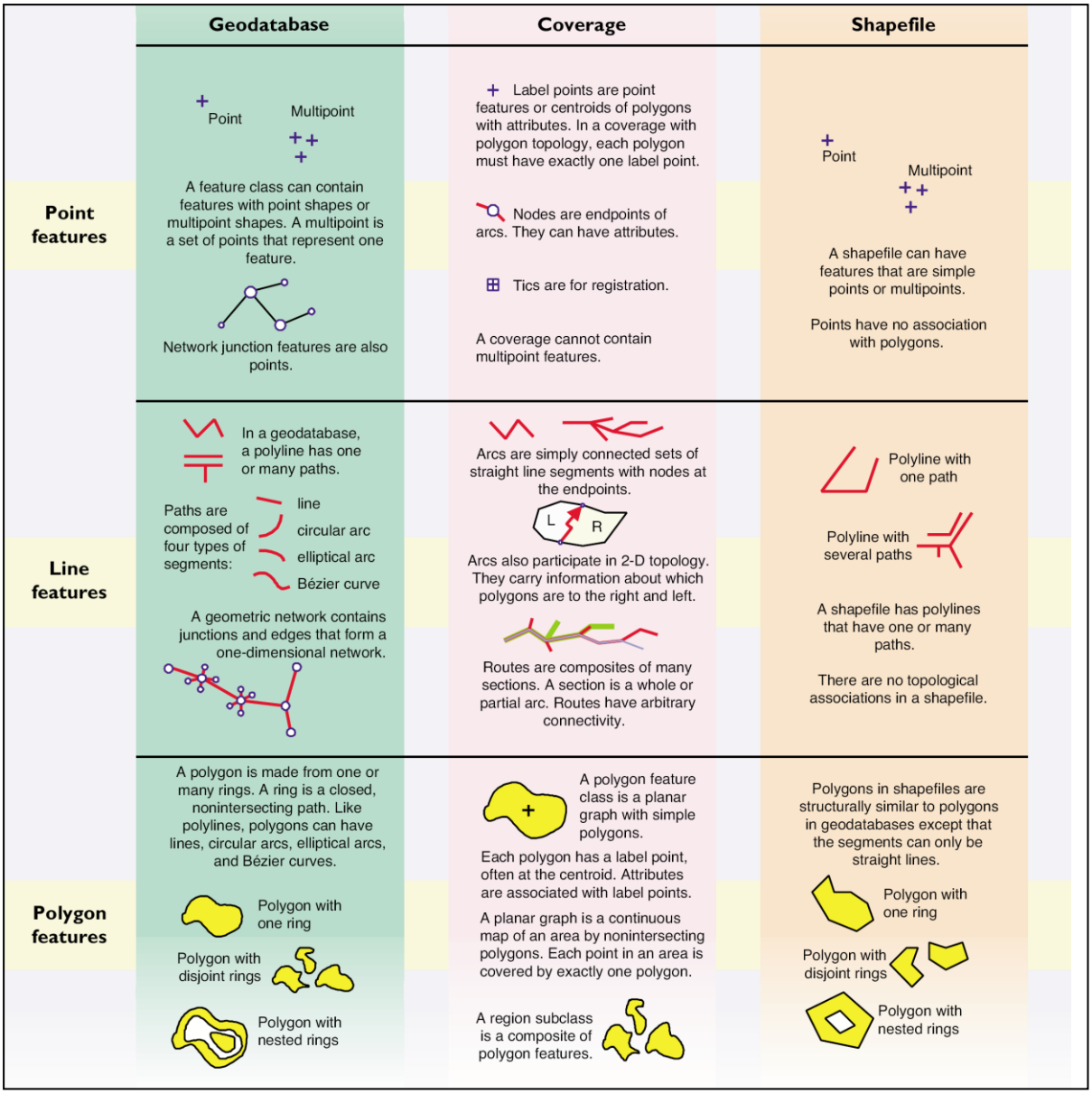


Figure 9.2 Differences between the three common data models which support vector data (Zeiler, 1999).

	DBMS	Client/Server	Objects	Long Transactions*	Editors	C or Java API	Raster	Size
Multiuser Geodatabases	Oracle, Microsoft SQL Server, Informix, IBM DB2	Yes	Yes	Yes	1 or more	Yes	Yes	Unlimited
Personal Geodatabase	Microsoft Jet	No	Yes	No	1 only	No	No	Up to 2 Gb

Figure 9.3 Properties of the multi-user and personal geodatabases (Zeiler, 1999)

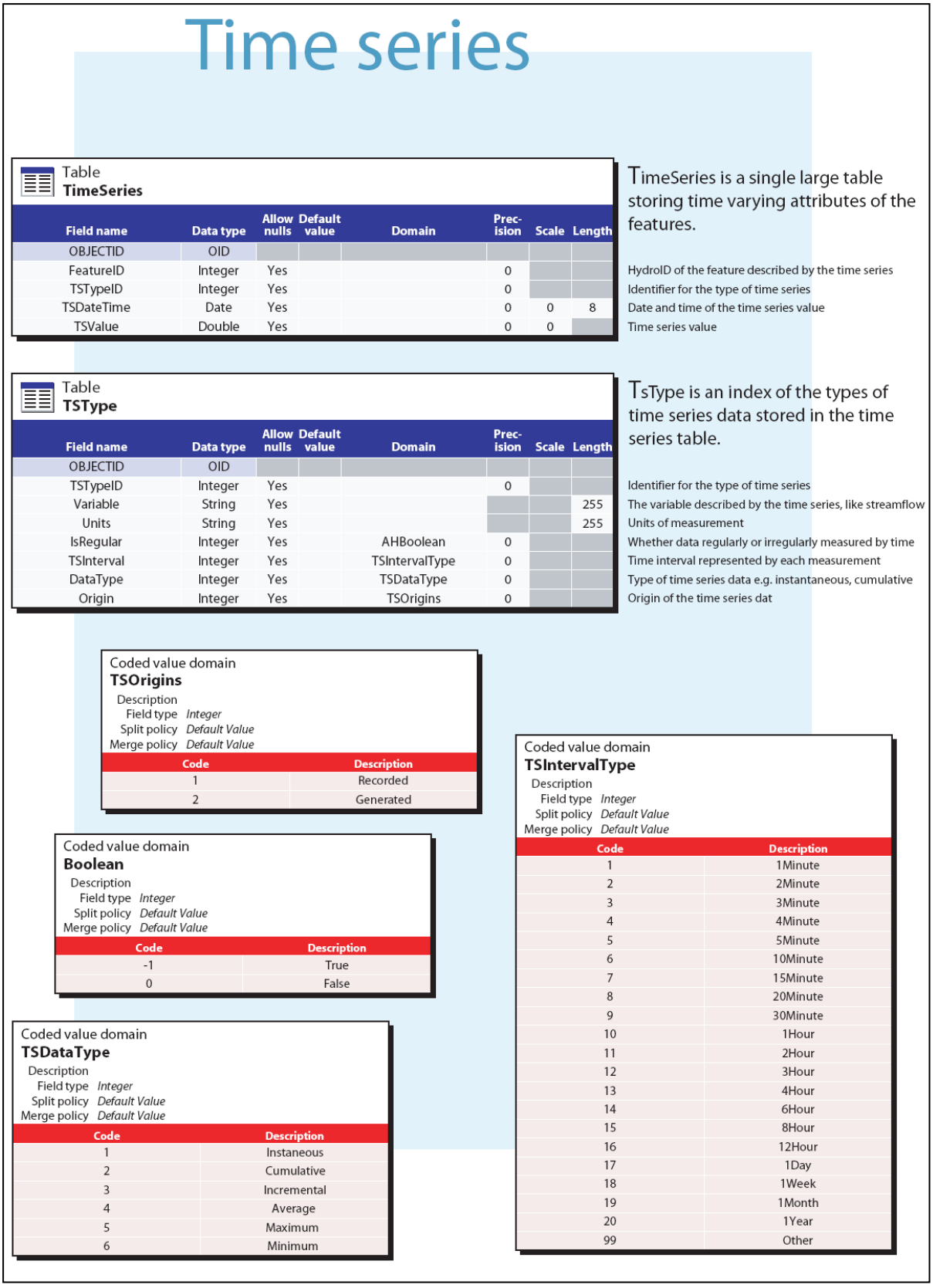


Figure 9.4 Time series component of the Arc Hydro data model (ESRI, 2003b)

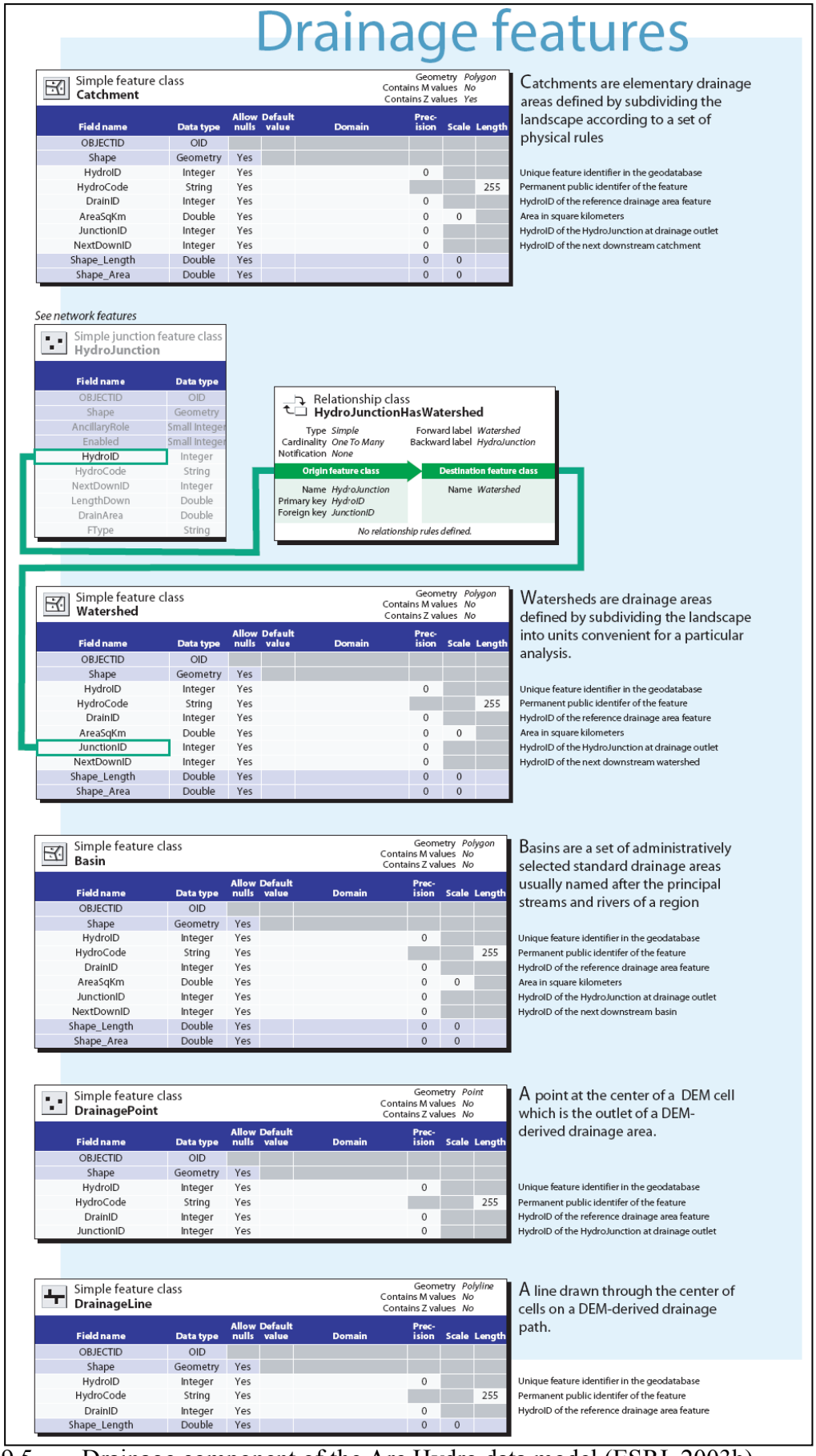


Figure 9.5 Drainage component of the Arc Hydro data model (ESRI, 2003b)

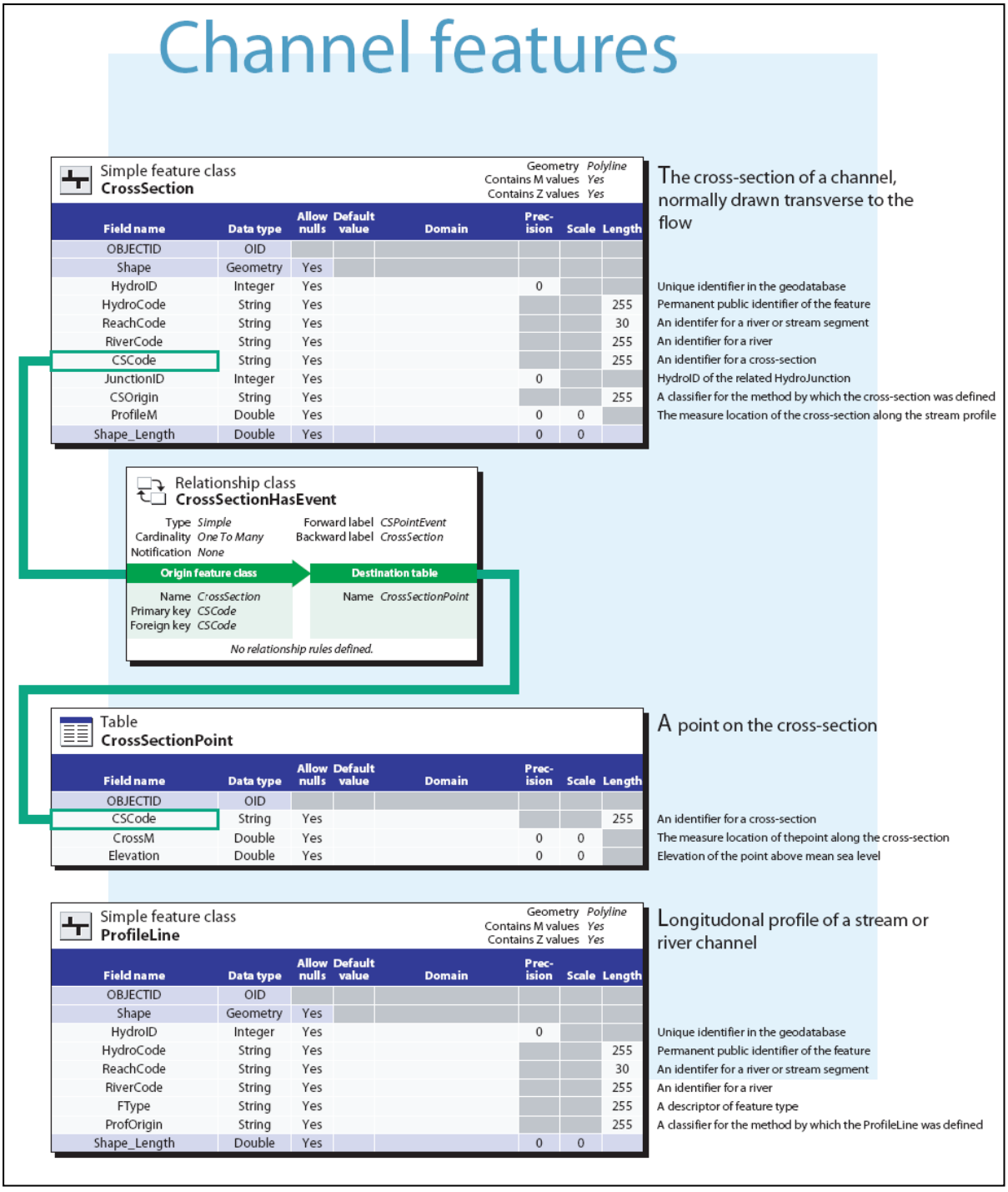


Figure 9.6 Channel component of the Arc Hydro data model (ESRI, 2003b)

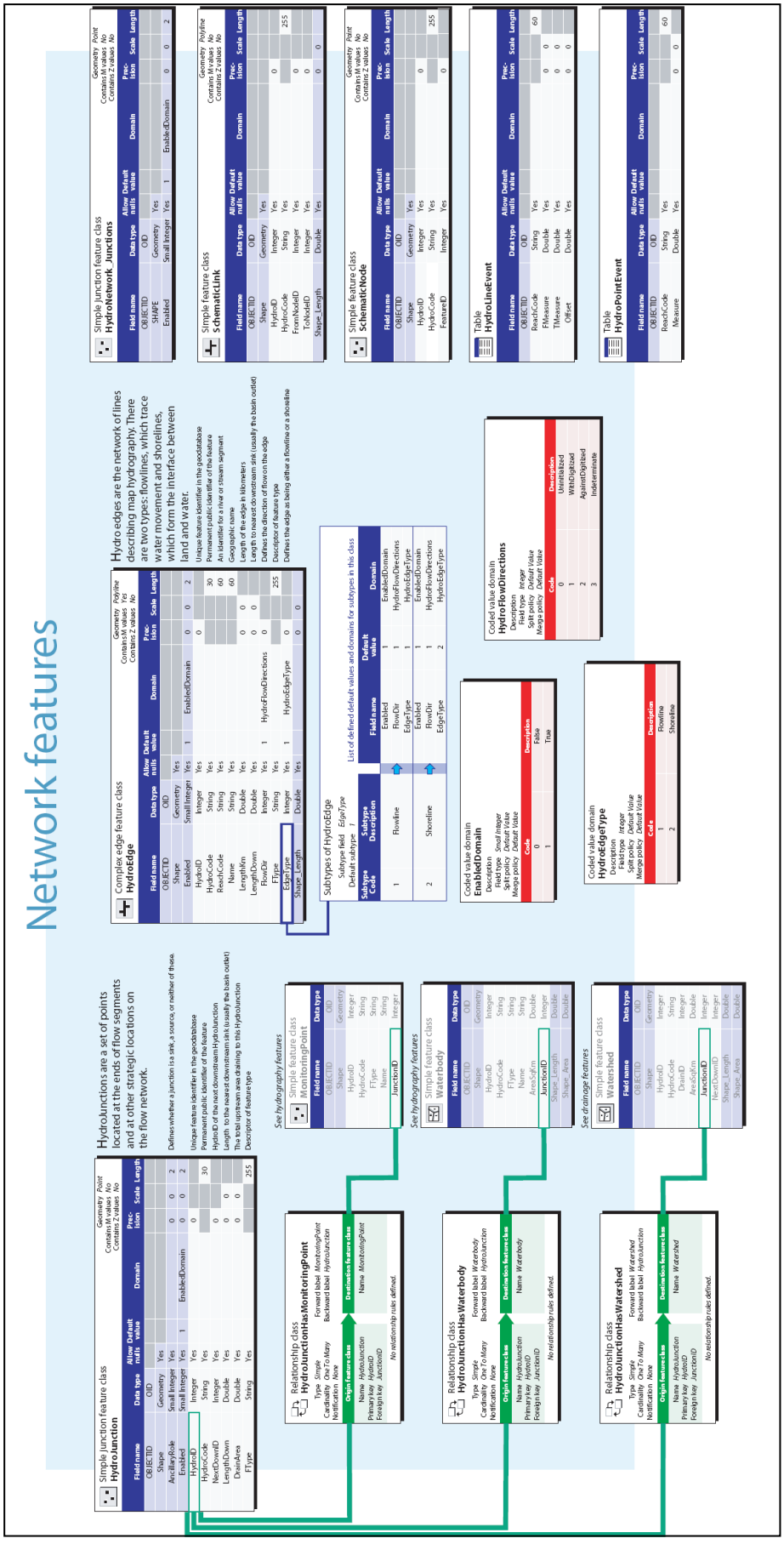


Figure 9.7 Network component of the Arc Hydro data model (ESRI, 2003b)

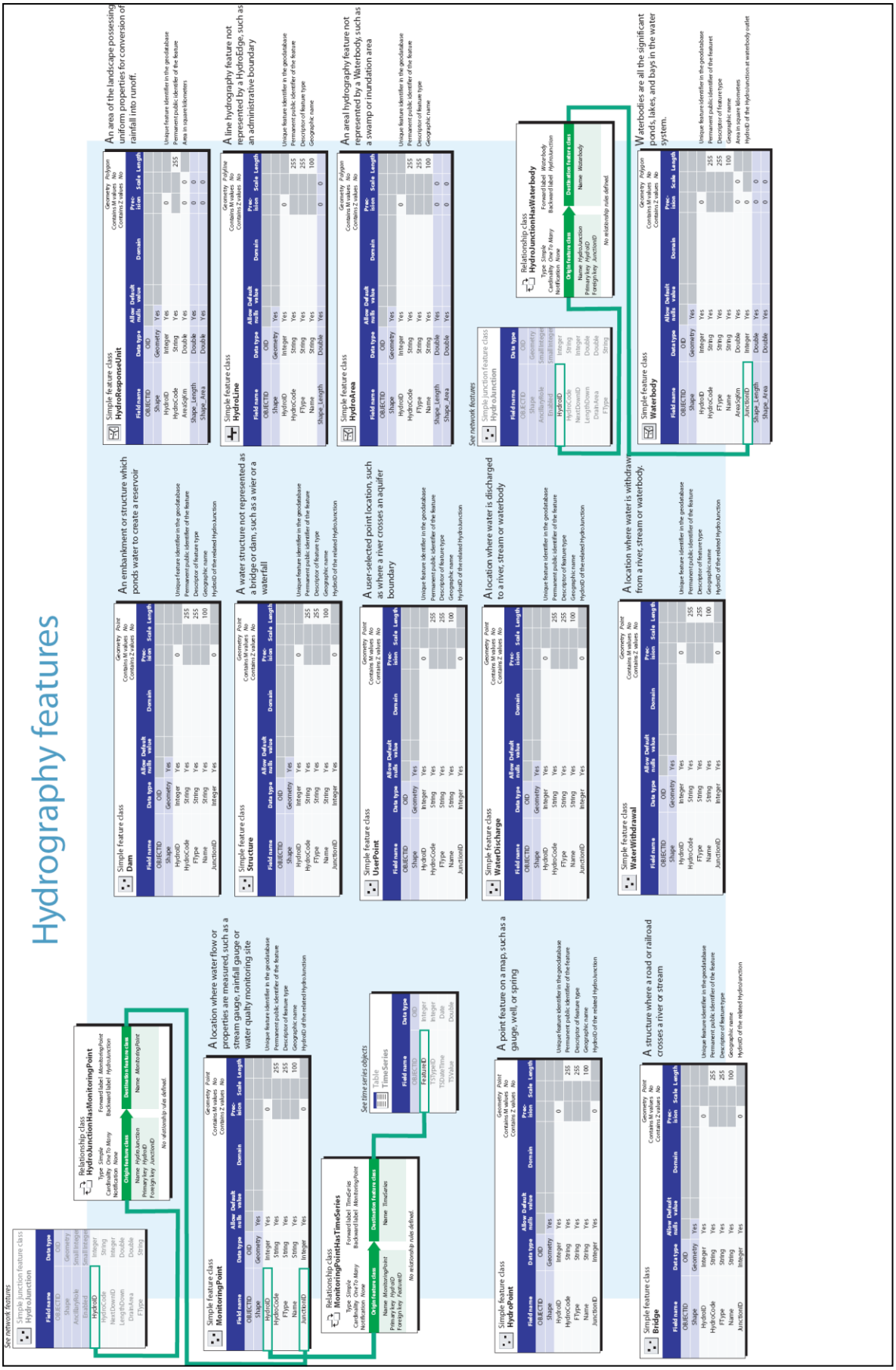


Figure 9.8 Hydrography component of the Arc Hydro data model (ESRI, 2003b)

```
Public Sub runQuery(strWhereClause As String) 'Visual Basic Implements a SQL Query on the Microsoft Access Database

Dim Myquery As String
Dim rs As ADODB.Recordset
Dim objAttribute As clsAtttribute
Dim objFeature As clsFeature

'This code is used to create the SQL clause
Myquery = Myquery & ""
Myquery = Myquery & "SELECT MTBL_AttributeData.FeatureID, MTBL_AttributeType.AttributeID,"
Myquery = Myquery & " IIf(MTBL_AttributeType.DataTypeTable='MTBL_ShortIntegerData',Str(MTBL_ShortIntegerData.Value),"
Myquery = Myquery & " IIf(MTBL_AttributeType.DataTypeTable='MTBL_LongIntegerData',Str(MTBL_LongIntegerData.Value),"
Myquery = Myquery & " IIf(MTBL_AttributeType.DataTypeTable='MTBL_TextData',MTBL_TextData.Value,"
Myquery = Myquery & " IIf(MTBL_AttributeType.DataTypeTable='MTBL_BooleanData',Str(IIf(MTBL_BooleanData.Value=0,0,1)),"
Myquery = Myquery & " IIf(MTBL_AttributeType.DataTypeTable='MTBL_DoubleData',Str(MTBL_DoubleData.Value),"
Myquery = Myquery & " IIf(MTBL_AttributeType.DataTypeTable='MTBL_DataTimeData',Str(MTBL_DateTimeData.Value),"
Myquery = Myquery & " IIf(MTBL_AttributeType.DataTypeTable='MTBL_SingleData',Str(MTBL_SingleData.Value),'error'))))))) AS MyQuery
"
Myquery = Myquery & " FROM (((((MTBL_LongIntegerData RIGHT JOIN (MTBL_ShortIntegerData"
Myquery = Myquery & " RIGHT JOIN (MTBL_AttributeData INNER JOIN MTBL_AttributeType"
Myquery = Myquery & " ON MTBL_AttributeData.AttributeType = MTBL_AttributeType.OBJECTID)"
Myquery = Myquery & " ON MTBL_ShortIntegerData.OBJECTID = MTBL_AttributeData.RecordInTable)"
Myquery = Myquery & " ON MTBL_LongIntegerData.OBJECTID = MTBL_AttributeData.RecordInTable) LEFT JOIN MTBL_TextData"
Myquery = Myquery & " ON MTBL_AttributeData.RecordInTable = MTBL_TextData.OBJECTID) LEFT JOIN MTBL_SingleData"
Myquery = Myquery & " ON MTBL_AttributeData.RecordInTable = MTBL_SingleData.OBJECTID) LEFT JOIN MTBL_BooleanData"
Myquery = Myquery & " ON MTBL_AttributeData.RecordInTable = MTBL_BooleanData.OBJECTID) LEFT JOIN MTBL_DateTimeData"
Myquery = Myquery & " ON MTBL_AttributeData.RecordInTable = MTBL_DateTimeData.OBJECTID) LEFT JOIN MTBL_DoubleData"
Myquery = Myquery & " ON MTBL_AttributeData.RecordInTable = MTBL_DoubleData.OBJECTID"

'If a string value has been passed to the procedure append it to the statement created above as the "where" part of the query
If Len(strWhereClause) > 0 Then
    Myquery = Myquery & strWhereClause
End If
```

Figure 9.9 A sample of the Visual Basic code used in to extract data from the Attribute Database, including much of the integrated SQL syntax