The Representation and Management of Evolving Features in Geospatial Databases

A thesis submitted by Alex Lohfink for the degree of Doctor of Philosophy.

Faculty of Advanced Technology,
Department of Computing and Maths, University of Glamorgan

April 3, 2009
DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed..................................................... (candidate)

Date .......................................................  

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

Signed..................................................... (candidate)

Date .......................................................  

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed..................................................... (candidate)

Date .......................................................  
Acknowledgments

I would like to thank the following people:

My supervisor, Tom Carnduff, for his advice, support, and guidance. My Director of Studies, Nathan Thomas, for his positive comments and direction. My second supervisor, Mark Ware, for his advice and encouragement.

I would also like to thank Glen Hart, Tony Joyce, and Mark Pendlington of Ordnance Survey for their support, encouragement, and hospitality.
Abstract

Geographic features change over time, this change being the result of some kind of event or occurrence. It has been a research challenge to represent this data in a manner that reflects human perception. Most database systems used in geographic information systems (GIS) are relational, and change is either captured by exhaustively storing all versions of data, or updates replace previous versions. This stems from the inherent difficulty of modelling geographic objects in relational tables. This difficulty is compounded when the necessary time dimension is introduced to model how those objects evolve. There is little doubt that the object-oriented (OO) paradigm holds significant advantages over the relational model when it comes to modelling real-world entities and spatial data, and it is argued that this contention is particularly true when it comes to spatio-temporal data. This thesis describes an object-oriented approach to the design of a conceptual model for representing spatio-temporal geographic data, called the Feature Evolution Model (FEM), based on states and events. The model was used to implement a spatio-temporal database management system in Oracle Spatial, and an interface prototype is described that was used to evaluate the system by enabling querying and visualisation.
# Contents

1 Introduction .................................................................................................................. 12
   1.1 The need to incorporate time into geographic data ............................................. 13
      1.1.1 Features of a temporal GIS ........................................................................ 13
      1.1.2 Applications of a temporal GIS ................................................................. 15
   1.2 The representation of spatial objects and time ..................................................... 16
      1.2.1 Hybrid and integrated models of space and time ........................................ 16
      1.2.2 Objects and events ..................................................................................... 18
      1.2.3 Data representation of multidimensional objects ....................................... 19
   1.3 Hypothesis and research objectives .................................................................. 20
      1.3.1 Hypothesis ............................................................................................... 20
      1.3.2 Objectives ............................................................................................... 21
   1.4 Structure of thesis ............................................................................................... 21

2 Spatial and spatio-temporal data modelling ............................................................. 23
   2.1 Managing spatial data ......................................................................................... 24
      2.1.1 Spatial data ............................................................................................... 24
      2.1.2 Spatial database management systems (SDBMS) .................................... 26
      2.1.3 Available SDBMS ..................................................................................... 27
   2.2 Spatial and spatio-temporal data modelling ....................................................... 31
      2.2.1 Modelling spatial data .............................................................................. 31
      2.2.2 Spatial and spatio-temporal extensions to the ER model ......................... 34
      2.2.3 Object-oriented geographic data modelling .............................................. 37
2.2.4 Temporal and spatio-temporal OO models . . . . . . . . . 41
2.3 Representing change and events in spatial data . . . . . . . . . 52
  2.3.1 Features and change . . . . . . . . . . . . . . . . . . . . 52
  2.3.2 Event-based spatio-temporal models . . . . . . . . . . . 53
2.4 Implementations of STDBMS . . . . . . . . . . . . . . . . . . . 67

3 The FEM spatio-temporal data model . . . . . . . . . . . . . . . 76
  3.1 The foundation of the FEM data model . . . . . . . . . . . . . 77
    3.1.1 Components of an optimal data model . . . . . . . . . . 77
    3.1.2 Basis of the design . . . . . . . . . . . . . . . . . . . . . 78
  3.2 FEM type descriptions . . . . . . . . . . . . . . . . . . . . . 82
    3.2.1 The spatial_object_state type . . . . . . . . . . . . . . 82
    3.2.2 The feature_state type . . . . . . . . . . . . . . . . . . . 83
    3.2.3 The event type . . . . . . . . . . . . . . . . . . . . . . . 83
    3.2.4 The transition type . . . . . . . . . . . . . . . . . . . . . 86
    3.2.5 The evolved_feature type . . . . . . . . . . . . . . . . . 87
  3.3 Modelling real-world data . . . . . . . . . . . . . . . . . . . . 87
    3.3.1 Alternative datasets . . . . . . . . . . . . . . . . . . . . 87
    3.3.2 ITN data . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
    3.3.3 Events in ITN data . . . . . . . . . . . . . . . . . . . . . 90
    3.3.4 Modelling ITN data with FEM . . . . . . . . . . . . . . . 93

4 Implementation of the FEM STDBMS . . . . . . . . . . . . . . . 104
  4.1 The benefits of the object-relational model . . . . . . . . . . 104
  4.2 Creating the base state . . . . . . . . . . . . . . . . . . . . . 105
    4.2.1 Implementing the FEM schema . . . . . . . . . . . . . . . 106
    4.2.2 Oracle objects and JDBC strongly typed interfaces . . . 107
    4.2.3 Creating FEM objects . . . . . . . . . . . . . . . . . . . . 108
  4.3 Updating the base state . . . . . . . . . . . . . . . . . . . . . 110
  4.4 Determining events . . . . . . . . . . . . . . . . . . . . . . . 113
    4.4.1 Event determination algorithm . . . . . . . . . . . . . . . 114
4.4.2 Spatial indexing .......................................... 116

5 Querying the FEM STDBMS ................................. 117
   5.1 Accessing spatio-temporal data ......................... 117
   5.2 Accessing FEM objects .................................. 119
   5.3 SQL 2003 and Oracle SQL ................................ 120
      5.3.1 Querying REFs ........................................ 121
      5.3.2 Querying subtypes .................................. 121
      5.3.3 Querying nested tables ................................ 122
      5.3.4 Temporal and spatio-temporal operators ............. 123
      5.3.5 Change expressions .................................. 124
   5.4 Example queries ....................................... 125
      5.4.1 Configuration-based queries ......................... 125
      5.4.2 Event-based queries .................................. 126
      5.4.3 Executing queries .................................... 128

6 Evaluation .................................................. 130
   6.1 Querying techniques .................................... 130
      6.1.1 Querying with SQL .................................. 131
      6.1.2 Querying using Java .................................. 131
   6.2 The FEM interface prototype ............................. 131
      6.2.1 Software and development environment ................ 132
      6.2.2 Requirements of the prototype ......................... 134
      6.2.3 Creating a base map with MapBuilder ................ 134
   6.3 Querying features by point-and-click .................. 136
      6.3.1 Retrieving configuration and chronicle data ........ 140
      6.3.2 Querying an evolved_feature object ................ 142
   6.4 Using dynamic themes ................................... 145
      6.4.1 Overlaying features .................................. 145
      6.4.2 Generating events .................................... 149
      6.4.3 Event-based spatio-temporal queries ................ 149
6.5 Query evaluation .............................................. 164
   6.5.1 Configuration-based queries ......................... 164
   6.5.2 Event-based queries .................................. 167
   6.5.3 Prototype evaluation ................................ 168
6.6 Limitations of the FEM system .......................... 169

7 Conclusions ...................................................... 171
   7.1 Proof of hypothesis ...................................... 171
   7.2 Contributions of this research ......................... 174
   7.3 Future work .............................................. 174

Bibliography ....................................................... 177
List of Figures

2.1 Oracle Spatial’s geometry types .............................................. 29
2.2 Development of a road bypass, taken from Worboys (1994b). 33
2.3 Temporal object and temporal instance properties (taken from Sotiropoulou et al. (1998)) .................................................... 43
2.4 TVM version derivation and attribute history (taken from Moro et al. (2002)) ...................................................... 44
2.5 TVM classes (taken from Moro et al. (2002)) .......................... 45
2.6 Versioned objects of the PagePattern class, showing one versioned object (Winter) (taken from Moro et al. (2002)) .......... 46
2.7 UML representations of a time variant property - Half-Hour Zone (taken from Price et al. (1999)) ................................. 47
2.8 Simple example of a GeoFrame-T model (taken from da Rocha et al. (2001)) .............................................................. 48
2.9 MADS schema diagram (taken from Parent et al. (1999)) ...... 50
2.10 The GEM model showing objects, events, and their interaction (taken from Worboys and Hornsby (2004)) .................. 55
2.11 Overview of the model described by Ikazaki and Watanabe (2006) .............................................................. 59
2.12 Evolution of a river boundary (taken from Galton (2004)) .... 59
2.13 History graph primitives (taken from Hornsby and Egenhofer (2000)) .............................................................. 64
2.14 History graph transitions (taken from Hornsby and Egenhofer (2000)) .......................... 65
2.15 STAO structure (taken from Raza et al. (1996)) ............... 71
2.16 Query results from the STAO test case (taken from [Raza et al. (1996)]). LU stands for Land Use. ......................... 72

3.1 Class diagram of the ITN FEM data model ......................... 81
3.2 ITN structure (taken from Ordnance Survey (2006)) ........... 89
3.3 Division event in ITN data .......................... 91
3.4 Fusion event in ITN data .......................... 92
3.5 Relocation event in ITN data .......................... 92
3.6 Redirection event in ITN data .......................... 93
3.7 Class diagram showing specialised classes for modelling ITN data ......................................................... 94
3.8 Object diagram modelling a road feature extended by the creation of two links in ITN data ....................... 97
3.9 Object diagram modelling a division event .......... 99
3.10 Object diagram modelling a fusion event ............ 101
3.11 Object diagram modelling redirection and relocation events ... 102

5.1 Data and search spaces of basic spatio-temporal queries(taken from Langran (1992)]). .......................... 118

6.1 MapViewer architecture, taken from Murray (2006). ........ 133
6.2 MapBuilder tool used to create a base map ........................ 135
6.3 Base map with MBR theme set as identifiable ........................ 137
6.4 Result of visually querying a feature .......................... 138
6.5 Result of visually searching and querying ........................ 139
6.6 Configuration for state 1 of the ‘Penbrey’ feature ............ 141
6.7 Event chronicle for state 2 of the ‘Penbrey’ feature ............ 143
6.8 Querying the evolved_feature object .......................... 144
6.9 Adding a dynamic theme .......................... 146
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.10</td>
<td>Overlay of states 1 and 2 of the ‘Penbrey’ feature</td>
</tr>
<tr>
<td>6.11</td>
<td>Figure 6.10 zoomed in to show differences between states</td>
</tr>
<tr>
<td>6.12</td>
<td>Base map enhanced with link objects of state number 2 within the area of the B4385</td>
</tr>
<tr>
<td>6.13</td>
<td>Figure 6.12 zoomed in to show differences between overlaid states</td>
</tr>
<tr>
<td>6.14</td>
<td>Dynamic themes panel showing the generation of JDBC theme to display events between two time</td>
</tr>
<tr>
<td>6.15</td>
<td>Base map enhanced with events using a dynamic theme</td>
</tr>
<tr>
<td>6.16</td>
<td>Figure 6.15 zoomed in showing creation events</td>
</tr>
<tr>
<td>6.17</td>
<td>Event overlay of states 1 and 2 of the ‘Penbrey’ feature - right end view</td>
</tr>
<tr>
<td>6.18</td>
<td>Event overlay of states 1 and 2 of the ‘Penbrey’ feature - centre view</td>
</tr>
<tr>
<td>6.19</td>
<td>Event overlay of states 1 and 2 of the ‘Penbrey’ feature - left side view</td>
</tr>
<tr>
<td>6.20</td>
<td>Events that have occurred between two times within a specified area</td>
</tr>
<tr>
<td>6.21</td>
<td>Figure 6.20 zoomed in showing a redirection event</td>
</tr>
<tr>
<td>6.22</td>
<td>Division events that have occurred between two times</td>
</tr>
<tr>
<td>6.23</td>
<td>Figure 6.22 zoomed in to a specific event</td>
</tr>
<tr>
<td>6.24</td>
<td>Redirection events between two times within a specified spatial region</td>
</tr>
<tr>
<td>6.25</td>
<td>Figure 6.24 zoomed in to show redirection events</td>
</tr>
</tbody>
</table>
List of Tables

2.1 The SDO_GEOMETRY data type ........................................... 28
2.2 Comparison between spatial and time dimensions ............... 53
2.3 Continuants in an object-based event model (taken from Galton (2004)) ......................................................... 61
2.4 Events in an object-based event model (taken from Galton (2004)) ......................................................... 62
3.1 Criteria for an optimal data model (taken from Connolly and Begg (2002)) ......................................................... 78
4.1 FEM object tables ......................................................... 106
5.1 SQL temporal operators .................................................. 123
Chapter 1

Introduction

At the heart of any geographic information system (GIS) is a database system. Data representing geographic entities or objects, and spatial features, as well as their attributes, are stored in these GIS and manipulated and visualised according to the user’s input. The rapid emergence of GIS has demanded the evolution of database systems to support these spatial data, and to provide powerful analysis operations and functions to assist in decision support, projections, predictions, and simulations in a wide variety of problem domains. Features in a GIS group together entities or areas that are of particular interest from a specific viewpoint, such as counties, population, forestation, or roads. Features change over time, this change being the result of some kind of event or occurrence, and there has been a great deal of research into representing this data in a manner that reflects human perception. Integrating some form of time into geographic data has, in general, been limited to retrieving a snapshot view of the study area, usually the most recent stored in the system. If change is captured, it is achieved by exhaustively storing all versions of data, or else updates are destructive, that is they replace previous versions (known as feature succession or feature substitution). Most GIS have relational database systems at their heart, and this presents an inherent difficulty when modelling geographic objects, particularly so when
the necessary temporal elements are introduced. Although some commercial
database management systems (DBMS) offer support for versioning data, im-
plementations of geographic spatio-temporal database management systems 
(STDBMS) have been limited due to the inherent complexity encountered
when reconciling a logical spatio-temporal representation within a relational
framework. There is little doubt that the object-oriented (OO) paradigm
holds significant advantages over the relational model when it comes to mod-
elling real-world entities. Indeed, the spatial elements of most current geo-
graphic database systems are represented as objects in an object-relational
model, and it is argued here that there is potential to extend these spatial
data types to be more representative and expressive, and to represent the
evolution of geographic features.

Research into spatio-temporal database systems can be broadly cate-
gorised into two areas: moving objects and evolving features. Moving ob-
jects research (for example Guting et al. (2000)) deals with the movement
of objects such as aircraft, people, and traffic, and implies a shorter time
granularity than that of evolving objects. Evolving features research deals
with either gradual change, such as environmental change, or discrete and in-
frequent change, such as occurs with administrative boundaries. This thesis
is concerned solely with evolving features.

1.1 The need to incorporate time into geographic
data

1.1.1 Features of a temporal GIS

It has long been established that the introduction of time and time-handling
capabilities into GIS is a necessary development [Langran (1992); Peuquet
(2003); Raper (2000); Galton (2004)] to enable the storage and recovery of
current, past, and future states of a particular study area.
A temporal GIS is a system that offers spatio-temporal capabilities to some degree. Although there is no definitive agreement on the range of functions that such a system should perform, Langran (1992) describes a possible set of functions that could be provided:

**Inventory**  A temporal GIS should be able to store the current state of a feature, as well as a complete history of its states representing its evolution over time.

**Analysis**  In a temporal GIS this would enable the understanding of processes representing change. This may be in the form of statistical analysis, identifying patterns in data, divergence or cross-referencing, or extrapolating data values or patterns from preceding known states.

**Scheduling**  Historical records (for example of road maintenance work) would help the scheduling of future work and deployment of resources. Thresholds could be set that when reached could trigger alerts for management. Advance scheduling might predict when thresholds would be reached, or even determine thresholds.

**Display**  This provides a visual interface to display current states and describe events and processes. Geospatial data can be displayed and data regarding events and states retrieved and displayed by interacting with maps via mouse clicks.

**Updates**  The most common disadvantage when updating data is that in the main it is destructive, that is the old data is deleted ( sometime referred to as change-only updates). For a system allowing the storage of spatio-temporal data, updates would be additions to expired or ‘non-current’ data, and would allow the retrieval of all past instances of the data, as well as operations on them.
Some hypothetical examples of how a temporal GIS could exploit these features are now described.

1.1.2 Applications of a temporal GIS

Forestation control  It would be advantageous in forest resource management to record the evolution of a plantation as a history of harvest cycles, including data concerning plantation, spraying, thinning, harvesting, selling, and rehabilitation. These histories would allow the projection of long-term timber yields, and growth rate comparisons over time, as well as help predict the spread of disease or fire [Langran (1992)]. Questions that the system might be able to answer are:

- Analysis - how has the forest cover changed in a study area over a given period of time?
- Prediction - based on data collected for a particular period, what is the probability of forest cover decreasing?
- Comparative analysis - how accurate were predictions of forest cover during this period compared to collected data?

[Medeiros and Jomier (1994)]

Infrastructure - water  A temporal GIS would allow a water utilities company to monitor public works and maintenance, keep track of works in progress, and extrapolate future requirements. A historical database could help detect weak links in a system. For example, in a water utility network, places with high instances of leakage could be identified, and scheduled for preventative maintenance, enabling efficient use of staff and equipment. Instances of leaks could be cross-referenced with other data to highlight any events that may be impinging on the integrity of pipe works, such as storms or floods.
Infrastructure - transport  As with a water utilities, spatio-temporal capabilities would enable maintenance, scheduling, and monitoring for transportation networks. The system can be supplied with road life expectancies, and updated with reports on the current condition so that maintenance work can be planned ahead. Accident rates could be explored and analysed, with reference to conditions such as the season, lighting conditions, and weather. Road network flows could potentially be modelled to predict the effects on traffic flows of these accidents, as well as that of road works and other events [Galton and Worboys (2005)].

1.2 The representation of spatial objects and time

Although at the simplest level the timestamping of geospatial tuples or temporal data in a database system constitutes the elevation of ‘standard’ data to spatio-temporal data, the organisation of the data, its structure, and its inter-relationships, must be considered to present the data in a meaningful and useful capacity. It is also pertinent at this time to distinguish between the two views used in GIS: the field-based view and the object-based view, as there are important differences between the two which impinge on their treatment of time. The field-based view ascribes attributes to locations, creating a continuous fabric [Galton (2004)], whereas objects are discrete units where location is a spatial attribute. Although this research is concerned solely with the object view, it was considered useful to include references to the field view for completeness.

1.2.1 Hybrid and integrated models of space and time

Enhancing objects with a temporal dimension transforms them from 2/3D representations to 3/4D, and raises questions about the nature of their repre-
sentation. Treating objects by the addition of a time attribute implies a 3D + 1 (or 3D + T) treatment, as opposed to a full-blown 4D approach. These approaches are described by Galton (2004) as 'space with time' and 'space-time' respectively. The space-time perspective treats temporal intervals and spatial regions as being represented as 'chunks' of space-time, occupied by material extents known as hyperobjects. Raper (2000) describes these approaches as 'hybrid' (3D + 1) and 'integrated' (4D). The hybrid approach treats space and time differently, and separates them in its representation. This implies linking spatial attributes with a time value, either a timestamp or time interval, within the structures involved in the representation. The integrated approach, on the other hand, suggests that the time dimension is integral with the three spatial dimensions, and, as Raper puts it, that such systems are defined by the spatio-temporal processes that transform entities. This view raises the question of how to represent true 4D objects.

In considering this, another distinction to be made is that between the absolute and relative models of time. In the absolute model, objects and events exist within a space-time framework, and are defined by their spatial and temporal extents within the boundaries of this framework. In the relative model, on the other hand, time itself is created by entities with a spatial and temporal extent, or as Galton (2004) puts it, objects and processes come first, with space and time being "logical constructs from the relationships among those entities".

This leaves several possible options for spatio-temporal representation. However, the inclusion of relative time in a representation is questionable, as it can be argued that relative time has negligible relevance to geography, and would also need an appropriate unit of measurement (Raper (2000) suggests light-seconds as a possibility). An example of an integrated approach using relative time is described by Raper and Livingstone (1995). In this approach, a system called OOgeomorph assembles four-dimensional entities from discrete points. The system uses classes to represent form, process, and
material, and the attributes of these classes are associated with a ‘time of knowing’, represented by a clock-time and date. Instances of these classes can be assembled into a ‘phenomenon’ object, the spatio-temporal structure of which depends on which attributes are selected. This model can be considered integrated in the sense that the 4D spatio-temporal structure of phenomena is constructed from objects and processes assembled into the phenomenon object, but the use of clock-time and date values suggests the use of absolute time, and is more akin to an integrated-absolute representation.

It would seem from this discussion, therefore, that, strictly speaking, all representations must be based on absolute time, and a degree of integration can be achieved, as with OOgeomorph. Other models can be developed as hybrid ‘space with time’ models using absolute time. An example of this type of representation is the ESTDM model Peuquet and Duan (1995). In this model an event represents a change between states, and the sequence of events through time is recorded in an ‘event list’.

1.2.2 Objects and events

It is argued in this thesis that an important theme that applies to any spatio-temporal model is the notion of events forming inter-relationships between objects in the space-time framework. Events are explicit ‘happenings’ in space and time, and from this perspective, objects in space and time make more sense. From this viewpoint, entities can be considered as continuants and occurrents. Continuants represent entities that persist through time, while occurrents represent events that happen, or occur, and are then gone. Examples of continuants are houses, people, and chairs. Examples of occurrents are lectures, car races, and a person’s life [Worboys (2005)]. Further, occurrents can be considered as discrete or continuous. For example, a discrete occurrent might be the time and date that a treaty was agreed, and be represented by a single date/time value, whereas a continuous occurrent would represent continuous change in an entity, such as the expansion of
a forest (an expansion event), and be represented by a time interval value. To distinguish between these two types of occurrent, the term ‘event’ will henceforth refer to discrete occurrents, and the term ‘process’ will refer to occurrents that represent continuous change.

If events and processes are represented, then an event-based view can be created, or as Galton (2004) describes, a chronicle-based view, where the histories of world entities are described by the events or processes that occur between states.

Alongside these considerations are the practicalities of representing multidimensional geographic objects in a database system.

1.2.3 Data representation of multidimensional objects

It can be seen, then, that the representation of multidimensional objects can be viewed from different perspectives, and involve varying levels or degrees of temporal enhancement. Further, any design and implementation of a spatio-temporal application must have a database system at its heart, consisting of static data values. Despite the discussion of objects and dynamic processes existing in space and time, and the various models for describing them, an effective implementation will require the storage of static data values to enable retrieval and manipulation. Even in a system that, for example, tracks a ships position via GPS and updates its location several thousands of times per second [Oracle (2004)], once a positional value is entered into the database system it is static. Certainly, dynamic functions and operations can derive other information or values from the static data. The challenge, then, is to organise and store these data in a manner that can provide some or all of the spatio-temporal functionality of a temporal GIS. An overview of some of the methods for achieving this is now given.

Histories The most obvious technique for representing evolving geographic objects is the time-slice snapshot [Langran (1992)], where successive states
of a feature are recorded in a temporally ordered list. There is, however, no way of determining changes which occur between snapshots, unless some method of interpolation can be employed. Further, there is no statement about how objects between recorded states came to change, that is, there is no way of handling temporal relationships between successive states. Snapshots are not a truly representative view of the world as changes do not occur in convenient, discrete increments, but as fragmented units or clusters of change, both related, inter-related, or unrelated to each other, occurring at different times. From this perspective, the snapshot representation is inadequate.

**Object mutation** Overcoming the limitations of the snapshot/histories model involves tracking the object as it mutates from one state to another through time whilst retaining its identity. One possible way to extend this concept is to define a set of change primitives which can then represent complex change. This kind of model has been described by Hornsby and Egenhofer (2000), where an object can undergo events such as creation, destruction, disappearance, reappearance, continuation, and death.

**Events and processes** A full-blown treatment of change requires the addition of events and processes as equal entities. In this model events are treated with equal status to objects, which are considered as continuants that persist through a period of change [Worboys (2005)]. This means that change to spatial objects and features is described by events as well as the relationships between them.

### 1.3 Hypothesis and research objectives

#### 1.3.1 Hypothesis

This thesis argues that a temporal GIS can be developed using standard object-oriented technology by employing an absolute-hybrid approach. This
involves the following:

- specifying an object-oriented conceptual schema in unified modelling language (UML), called the Feature Evolution Model (FEM), to effectively represent complex geographic objects as they evolve over time, and the events that transform them

- providing an implementation of the FEM model to demonstrate visualisation and querying of the FEM spatio-temporal database management system.

1.3.2 Objectives

The following objectives will need to be achieved in order to prove the hypothesis:

1. To use object-oriented design techniques to develop a conceptual data model for the representation of spatio-temporal geospatial data.

2. To develop queries for retrieving and analysing spatio-temporal geospatial data.

3. To implement an object-oriented temporal GIS based on the conceptual data model to enable visualisation and querying of spatio-temporal geographic data.

4. To test and evaluate the data model and the implemented system.

1.4 Structure of thesis

The rest of this thesis is organised as follows: the next chapter will review methods for managing and representing evolving geospatial data; chapter 3 will describe the development of the FEM spatio-temporal data model, its main types, and demonstrate how it can model real-world data; chapter 4
will describe the implementation of a FEM spatio-temporal database management system (STDBMS); chapter 5 will describe the benefits and methods for accessing data in the FEM STDBMS and provide some example queries; chapter 6 will discuss the evaluation of the application, and finally in chapter 7 the conclusions will be presented.
Chapter 2

Spatial and spatio-temporal data modelling

The aim of this chapter is to introduce the methods and concepts behind spatial and spatio-temporal data modelling. Before these topics are discussed, a little background into spatial data handling will be given. In section 2.2 methods of spatial and spatio-temporal data modelling will be compared and contrasted as to their effectiveness, with particular attention to highlighting the benefits of the OO approach. This section will include the critical discourse on the introduction of time and the various methods for data representation this entails, along with the inherent problems that can be encountered when storing such data in a STDBMS. Section 2.3 will look at models for the representation of change in object-based spatio-temporal data, and finally section 2.4 will describe some implementations of STDBMS, both prototypes and working examples.
2.1 Managing spatial data

2.1.1 Spatial data

Spatial information is such that location has some importance or relevance, and is not necessarily concerned with points on the earth’s surface. The term spatial can refer to any multi-dimensional system, such as medical images (referencing the human body), architectural drawings (referencing buildings), or engineering drawings (referencing mechanical objects). Geospatial data is a subset of spatial data, and refers to the surface of the earth, the oceans, and the atmosphere. Geospatial information contains locational data either as an explicit geographic reference, such as longitude and latitude, or implicit, such as address data. Some systems provide a mechanism to determine explicit references from implicit ones, known as geocoding. For GIS purposes, geographic data has four characteristics: its geographic position (coordinates), its attributes (data values), its topological relationships, and its time components. Once stored in a database, this data can be classified into three categories:

- conventional data - alphanumeric attributes
- spatial data - describes the geometry and location of geographic entities, and can contain topological information
- pictorial data - images that can be manipulated by image processing functions.

[Medeiros et al. (1994)]

Henceforth, the term spatial will refer to geospatial data.

There are two fundamental data models used to represent geographic entities: the raster data model, sometimes referred to as the field view, and the vector model, referred to as the object view. Raster data models define the world as a regular set of cells in a grid pattern, with values attached
to each cell. Raster is the natural way to represent continuous features, for example elevation, precipitation, slope, forestation, and habitat. These features usually show continuous change over wide areas. Raster cells have a characteristic cell dimension or cell size, and the cell dimension is important: a small reduction in the cell size gives a correspondingly large increase in the number of cells required to cover a given area: the number of extra cells is the square of the reduction in cell size, so cutting the cell dimension in half gives a four-fold increase in the number of cells. Each raster cell represents a given area on the ground, and is given a value that applies to the whole cell [Bolstad (2002)].

In a vector model, objects are modelled using sets of coordinates to represent their geometric shape. In this way, geographic features like locations of customers, delivery sites, buildings, and counties, can be modelled as two-dimensional objects formed from connected points. Objects such as towns or places of interest can be represented as points, streets and rivers can be represented as line strings, and shapes representing buildings or city boundaries can be modelled using polygons [Kothuri (2005)]. Vector data can include topological data, such as connectivity and adjacency, which aids spatial analysis. Early vector models did not maintain references to topological data, giving rise to the so-called spaghetti model.

Geographic objects exist in space, and as such are often referred to as spatial objects. Spatial objects are defined according to the space in which they are embedded, for example Euclidean space, metric space, or topological space. The most common underlying space for a spatial object is Euclidean space, in that spatial objects are typically represented by sequences of Cartesian coordinates, which determine each point in a plane by two numbers, usually called the x and y coordinates. Spatial objects must allow analysis operations to be performed on them by geometric algorithms, such as intersection and buffer (where an area is created around a geometric object to create, for example, a sales region), and therefore objects must be dis-
cretised in order to make them suitable for such analysis. Spatial objects are discretised in a vector model as point, line, and polygon geometries. In a spatial database management system (SDBMS), spatial objects typically need to represent location, thematic attributes (for example name, ownership, address), and temporal extent to record the impact of change [Worboys and Duckham. (2004)].

Raster data is generally simpler, is good for representing changing features, and is the most practical method for storing digital imagery. Vector data, on the other hand, offers more compact data storage, is easier for presenting maps in a preferred format, and is the most efficient way of capturing and representing networks and topological data. This project is concerned only with vector data.

2.1.2 Spatial database management systems (SDBMS)

The term “spatial database system” is in widespread use and refers to a view of a database system which contains sets of spatial objects, as opposed to other, aspatial objects or tuples of aspatial data. According to Guting (1994), a spatial database system must be a database system which “offers spatial data types in its data model and query language” and “supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial join”. The information contained in the spatial database is held in the form of digital co-ordinates, which describe the spatial features. As stated earlier, these can be points (for example, hospitals), lines (for example, roads), or polygons (for example, administrative districts). Normally, the different sets of data will be held as separate layers, which can be combined in a number of different ways for analysis or map production.
Object-relational database management system (ORDBMS)

Because of the success of relational database management systems (RDBMS), many such systems have been extended to include OO features. These systems are relational in nature as they support SQL (see section 5.3), but they also have OO features in that they support complex data types [Stonebraker (1996)]. This merging of the relational and OO models is known as the object-relational (OR) model. Many of the SDBMS described next utilise the OR model due to the need to represent spatial data.

2.1.3 Available SDBMS

Oracle Spatial

Oracle Spatial [Murray (2005)] exploits Oracle’s object-relational features to represent spatial objects as abstract data types (ADTs). More detail on these features is provided in section 4.1. Oracle Spatial provides many powerful functions and operators to manipulate spatial data, and includes support for raster and vector data models, a topology data model, quadtree and R-tree spatial indexing, query processing, and geocoding to location-enable existing business data stored in Oracle tables. Oracle Spatial represents spatial objects using its SDO_GEOMETRY data type. As Oracle Spatial will be used for the implementation phase of this research, details of this data type will now be given. Table 2.1 gives a description of the SDO_GEOMETRY data type.

There follows a description of each attribute of the type.

**SDO_GTYPE**

This attribute determines the geometry type represented by the object, and is a four digit number.

**SDO_SRID**
Table 2.1: The SDO_GEOMETRY data type

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDO_GTYPE</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SDO_SRID</td>
<td>NUMBER</td>
</tr>
<tr>
<td>SDO_POINT</td>
<td>SDO_POINT_TYPE</td>
</tr>
<tr>
<td>SDO_ELEM_INFO</td>
<td>SDO_ELEM_INFO_ARRAY</td>
</tr>
<tr>
<td>SDO_ORDINATES</td>
<td>SDO_ORDINATE_ARRAY</td>
</tr>
</tbody>
</table>

This specifies the coordinate system to be used by the geometry, and can be a number, or null. A null value will mean no coordinate system will be used. If the value is not null, then it must correspond to a value in the metadata table associated with its layer.

**SDO_POINT**

This is defined by the SDO_POINT_TYPE object type. If this value is not null, and the values of SDO_ELEM_INFO and SDO_ORDINATES are null, then it will be considered a point geometry. For any other scenario, it will be ignored.

**SDO_ORDINATES**

This stores a sequence of coordinates for all the elements of the geometry.

**SDO_ELEM_INFO**

This attribute specifies the type, ordering, and connection (that is by line or arcs) of geometries in the SDO_ORDINATES array.

Spatial supports primitive geometry types, and collections aggregated from these primitives. Geometry types supported by Spatial are:

- Points and point clusters
- Line strings
- n-point polygons
Figure 2.1 shows the geometry types supported by Spatial.

**DB2 Spatial Extender**

The Spatial Extender [IBM (2006)] is an extension to the IBM DB2 Universal Database. As with Oracle Spatial, it manages complex geographic data types alongside relational data in tables. The system is compatible with the ISO SQL/MM [ISO (2002)] spatial standard, and the OpenGIS Consortium’s Simple Features Specification [Open GIS Simple Features Specification for
SQL Revision (1998)]. The extension can also integrate with third party GIS software such as ESRI's ArcGIS.

**Informix Spatial Datablade**

The Informix Spatial Datablade [IBM (2005)] is an extension to the IBM Informix Dynamic Server (IDS), and like Oracle Spatial is object-relational. As with the DB2 extension, it supports SQL spatial queries and the OpenGIS Simple Features Specification, and has a multidimensional R-tree index to facilitate query processing. Support is also provided for time-series and temporal data, integration with GIS software, and has a binding to the C++ language to facilitate application development.

**PostGIS**

PostGIS [PostGIS (2008)] is a spatial extension to the the open source object-relational database PostgreSQL [PostgreSQL (2008)]. It provides support for programming APIs (including Java) and complies with the OpenGIS Simple Features Specification.

**ArcGIS**

Although ArcGIS [ESRI (2006)] is not a database system but rather a powerful GIS application, at its heart is a spatial database. ArcGIS supports four geographic data types: vector, raster, triangulated irregular networks (TINS), and tabular data. It uses a geodatabase as its core storage mechanism for spatial and attribute data, and this provides the “primary data format used for editing and data management” [ESRI (2008)]. The geodatabase stores spatial data in tables with spatial objects in table columns, and this structure allows ArcGIS to interface with other spatial database systems, such as Oracle Spatial, via ArcSDE (Spatial Data Engine [ESRI (2005)]).
ArcGIS also stores vector data in the form of coverages, which can contain topological information, or in shapefiles, which contain location and attribute information about features, but not topological information.

2.2 Spatial and spatio-temporal data modelling

2.2.1 Modelling spatial data

Spatial database systems are a special case of information systems in that the contained data has position in space. In considering the modelling of geospatial data, the following points are of particular importance [Hadzilacos and Tryfona (1998)]:

- An object’s position. Real-world objects are located in space, and for geospatial data we are interested in an object’s position, shape, orientation, and size.

- Different views of objects. Geospatial objects can be represented at different scales and resolutions, for example a town could be represented as a polygon or a point.

- Space-dependent attributes. This is a basic aspect of geospatial data. Some spatial attributes do not belong to any particular object, for example soil type is not specific to a particular land parcel. If the land parcel moves, it loses the soil type attribute and takes on another attribute from its new position. These are called space-dependent attributes.

- Spatial relationships. Relationships between geographic objects are dependent on their relative positions in space. These relationships include distance, connectivity (topology), directional relationships, and aggregation/composition relationships. A geographic data model should model these relationships in the form of integrity constraints.
• Complex objects. Complex spatial objects differ from aggregated non-spatial objects in that the spatial dimension is significant.

• Access and retrieval of geographic data. This includes accessing spatial objects along with attribute data and layers. This is achieved using methods, of which there are three types:

  1. Those that manipulate objects and operate only on attributes (for example changing the name of a feature).
  2. Spatial operations, for example retrieve the distance between two land parcels.
  3. Those which operate on layers, for example overlay soil type and erosion on a map.

**Transaction and valid time**

When considering the inclusion of time into a data model, at least two temporal dimensions must be considered: transaction time and valid time. Transaction time, also known as database time, is the time at which the data is entered into the database, whereas valid time, or event time, represents the real-world time of the representation. If these two temporal values are to be considered then temporal database systems can be divided into four categories: static, historical, rollback, and bi-temporal [Snodgrass (1992)]. Each of these supports zero to two of these temporal dimensions. Static systems represent neither dimension, historical systems support valid time only, rollback systems support transaction time only, and bi-temporal systems support both time dimensions. For geographic data, situations could arise where the recording of valid and transaction times might be beneficial. Consider the three scenarios depicted in 2.2 (described by Worboys (1994b)). Here, data exists describing these scenarios.

  • State 1, June 2003: the diagram shows a road passing through a town, and a proposed bypass to be built by December 2003.
Figure 2.2: Development of a road bypass, taken from Worboys (1994b).

- State 2, January 2004: here the diagram shows the resultant bypass, with a spatial configuration different to the 2003 projection, perhaps due to diversion around, for example, a conservation area.

- State 3, January 2005: this diagram shows another alteration, depicting that a section of the bypass was not actually built until 2005.

Data is available for all these scenarios. If only transaction time is recorded, and we assume a negligible time-lag between the reported events and their entry into the system, then the data for each state will be entered into the database at their corresponding times as transaction times, with no time specification for the projected bypass configurations, or their completion dates. Clearly these scenarios require the recording of the time when the configuration of the bypass either was valid, or was projected to be valid. If these times are also entered, then in 2005 we can perform the query ‘What was the spatial configuration of the bypass as forecast in June 2003?’. This query requires the recording of both transaction time (June 2003) and valid time (December 2003).

The value of this type of query set against the increased complexity of storing both types of time is a key consideration in the development of a temporal GIS.

Spatial and spatio-temporal data can be modelled conceptually using the entity-relationship (ER) model or the OO model. There have been some
attempts at extending both modelling techniques to encompass the special concerns of spatial and spatio-temporal data. Some extensions to the ER model will now be described.

2.2.2 Spatial and spatio-temporal extensions to the ER model

Hadzilacos and Tryfona (1997) describe Geo-ER, an extension of the ER model that models geographic data at the conceptual level. Geo-ER extends the ER model with special entity sets and relationships to model geographic entities’ positions, their space-dependent attributes, and their spatial relationships. It adds two special constructs: spatial aggregation, and spatial grouping. The space entity set is introduced, and entity sets of spatial objects are called geographic entity sets. As positions are a subset of space, then the entity set space is a power set of positions. Shape is represented by entity sets of 0, 1, and 2 dimensions, that are called geometric types. At this level of modelling, spatial relationships, dimension, and space-dependent attributes are represented, but the model does not seem to be a natural way to model spatial data, and no temporal dimension is supported.

The ER model is further extended by a model called the spatio-temporal relational data model (STRM) [Tryfona (1998)], to include temporal attributes, and includes both valid time and transaction time. Here, geographic objects and layers are defined as relations, objects having a geometry type with spatio-temporal attributes, and layers that take objects as foreign keys and store space-dependent attributes. Both objects and layers have a valid time attribute and a transaction time attribute. This model makes use of the implementation of the ubiquitous relational model and adds some key features necessary to model spatial-temporal data, including the definition of object relations to represent geographic objects, which moves it toward an object-relational model. This means that the model could extend an existing relational system, and the inclusion of temporal attributes means that evolv-
ing features could potentially be handled. However, aggregation of objects into features (layers) is simply represented by another relation, which is not a natural way of modelling the feature, and geometries are only described at a basic level (that is by name and dimensions, not coordinates) meaning that only attributes relating to the spatial object can be modelled.

Tryfona and Jensen (2000) describe another extension to the ER model that represents commonly modelled scenarios as abstracted modelling units, and includes spatial, temporal, and spatio-temporal extents. This model offers support for point, line, and polygon geometries, spatial location, modelled as an is_located_at relationship, and temporal extents. The supported temporal extents are existence time, valid time, and transaction time. This extension uses abstractions to reduce the inherent complexity in an ER diagram caused by introducing geometric entities and time. The method successfully simplifies the diagrams, and again moves the relational model towards an OO approach by encapsulating data into a higher level abstraction.

Another example of an extended ER model is the Geographic Information System Entity Relationship model (GISER) [Shekhar et al. (1997)]. The GISER model is based on four main concepts: space/time, features, coverages, and spatial objects. Space/time models relative positions and directions, and represents multidimensional extents in which geographic entities can exist and events can occur. This is a continuous field and can be discretised into, for example, calendars. Features model geographic phenomena (rivers, buildings, land parcels etc.), and these are modelled as continuous fields that change in relation to space/time. Discretisation of features forms coverages (equivalent to layers) that are stored in the database. In this way, a feature may be represented by a number of coverages representing the feature at different resolutions and accuracy. A coverage consists of spatial objects that possess geometric and/or topological properties, and exist in space/time.

These attempts at extending the ER model all offer the potential to model
geographic data within an existing relational database system, and GISER, STRM, and the *modelling units* models described here model spatio-temporal data. The benefit of these spatio-temporal models is their compatibility with the relational model, and this makes them attractive propositions. Geo-ER does not offer the necessary time dimension, but it shows how the ER model can be used to represent more complex objects by providing higher levels of abstraction. This concept is further developed by *modelling units*, where entities are grouped together to form reusable groups, and this is a necessary abstraction as the ER model becomes cluttered when dealing with the complexities of spatio-temporal data. However, this technique seems to be hiding the inherent complexity, making the model more manageable and comprehensible by database designers, rather than tackling this complexity at its heart, and if put into the context of an implementation, the benefits become less distinct. The GISER model does not attempt to simplify its representation with abstraction but rather tries to model field and object based spatio-temporal data by adding extra symbols to represent spatial attributes, relationships, and transformations. This model certainly has the necessary scope in its modelling capability, and tackles the object-field dichotomy, but again at the cost of complexity. Also, the additional symbols make comprehending the diagrams much more difficult.

These models show that the ER model can provide abstractions to represent spatio-temporal data, but incur the increased complexity required by the representation of geometric objects, their aggregation into layers, and the need to represent temporal attributes. This is in part due to the limitations of the relational model, which does not naturally model complex objects. It is the contention of this research that objects offer the necessary abstraction capabilities to represent spatio-temporal data in a more natural and expressive way, and in the next section, object-oriented techniques will be discussed.
2.2.3 Object-oriented geographic data modelling

RDBMS are widespread and hugely successful in modelling many applications that require the storage and manipulation of business and personal data. However, they have been shown to be limited when dealing with complex objects that represent more real world entities, and as mentioned earlier, spatial objects fall into this category. The object-oriented (OO) model has also been shown to be much more successful in representing complex design objects in the fields of CAD/CAM, software engineering, and other design environments, and although historically GIS has been developed with the prevalent RDBMS, its feature-based structure means that it lends itself naturally to the object-oriented paradigm [Worboys et al. (1990)]. The fundamental concept of object-orientation is that of the human perception of the world being composed of objects, which interact with one another. It is clear that object-based (vector) geographic data is more naturally modelled using object-orientation. A geographic object’s description is provided by static properties (for example a city name), behavioural characteristics (for example calculating the city’s representation at a different scale), and structural aspects (that is placing the object into its spatial context) [Worboys (1994a)]. Some of the key features of object-orientation and how they apply to geographic data are now described.

**Identity** Object identity (oid) provides a unique identity independent of other attributes of the object. The oid is never altered, and can only be removed when the object is dropped or destroyed.

**Encapsulation** This is a principal feature of object-orientation, and aims to make a software object reusable by shielding its internal data and operations from other objects. Data, attributes, and operations are all defined in the class or type, and contained within all instances of the object.
Classification or typing This refers to the mapping of object instances to a common class or type. Every object is an instance of at least one class, therefore this mapping corresponds to an ‘instance_of’ relationship, and applies to objects, types, and abstract data types (ADTs, see section 4.1) [Egenhofer and Frank (1992)].

Association An association is a relationship between two objects. This is a particularly expressive feature of OO, where associations can be explicitly modelled using appropriate semantics, rather than implying the association using relations, as in the ER model. Attributes can not only be data values, but also other objects or collections of objects. This can also include temporal attributes, meaning, for example, if a spatial object was mutated by an event object, this could be represented by a ‘mutated_by’ association linking the two objects. This would be implemented by way of an object pointer, or reference, and provides navigational access to referenced objects as opposed to the associative access of the relational model. Special cases of associations are inheritance and aggregation, which will be described next.

Inheritance Inheritance allows common attributes and operations from objects to be factored out into a superclass. This greatly reduces data redundancy within a system, and facilitates updates and maintenance. A class with additional attributes and/or operations derived from the superclass is called a subclass, and this relationship is often referred to as a parent/child relationship. A subclass can itself be a superclass to its own child class. Thus the class ‘Building’ could have a subclass ‘Residence’ which in turn could have its own subclass ‘RuralResidence’. Inheritance relationships are often termed ‘is a kind of’ relationships, although this is not entirely accurate as both subclasses and superclasses are abstractions of the same object. Inheritance is an efficient means of modelling sets of classes in a GIS, as typical objects such as roads, rivers, and buildings, along with their corresponding attributes and operations, can be defined in a high-level class and inherited
to the GIS application. For example, the superclass \emph{GEOMETRIC} could
define geometries, with location and spatial operations like nearest neigh-
bour, intersection, inclusion, distance and area. A class \emph{BUILDING} could
be described as a subclass of \emph{GEOMETRIC}, inheriting all its properties.

\textbf{Aggregation} \hspace{1em} This relationship models objects that are themselves com-
posed of other objects, and is referred to as an ‘is_part_of’ relationship.
For example, a city may be composed of house lots, streets, and parks.

\textbf{Versioning} \hspace{1em} The concept of representing changing real-world phenomena
over time raises the issues of versioning and configuration management. As
features change over time, their representation in the database must be mod-
ified, and this can be reflected by the creation of new versions [Wachowicz
and Healey (1994)]. Object-oriented data models offer substantial potential
support for versions, and versioning in geospatial database systems offers
the prospect of being able to handle the management of evolving features.
There are two main approaches to this. The first attaches a version number
and/or timestamp to each attribute of the object, providing a historically
ordered list. The second is to track change at the object level, generating
a new identifier for each new version, and linking objects with parent/child
relationships.

The versioning of objects in database systems [Sciore (1994); Golendziner
and dos Santos (1995)] has been developed as part of the design engineering
environment in CAD systems and represents the evolution of complex design
artifacts (composite objects)[Katz (1990)]. Here versions can be defined both
at the component level and at the composite object level. The proliferation of
configurations of composite objects in CAD versioning has led to the develop-
ment of versioning models that represent a configuration as a generic object,
containing no specific versions of its components. The generic object stores
references to its components’ types, and specific versions are resolved at run-
time by a process called Dynamic Reference Resolution (DRR)[Katz (1990);
Dittrich and Lorie (1988); Chou and Kim (1988)]. DRR thus minimises version percolation, where an update to a component triggers cascading updates to the configurations, and allows the free combination of components to create new configurations. Although there are parallels between versioning CAD objects and geographic objects, the two are conceptually different. In CAD, versioning facilitates the design process and allows multiple participants in this process. Designs are related to each other by derivation hierarchies composed of parent/child relationships. Conceptually it is more natural to refer to successive versions of geographic objects as states. States are not related by a derived from relationship but by the events or processes that created them. Another parallel with CAD is that of the versioning of composite objects. Geographic features can be composed of multiple geometric objects, but their configuration is relatively limited, negating the benefits of dynamic referencing. The principle of states and events, however, remains appropriate to component geographic objects.

OO modelling can be performed using the unified modelling language (UML) where classes and their attributes are denoted diagrammatically along with the associations between them. The combination of encapsulation, collection attributes, and object associations provides a powerful mechanism for data representation. Despite this, some attempts have been made to enhance UML for the representation of spatial data. One such example is called Geo-OM (Geographic Object Model) [Tryfona et al. (1997)]. Geo-OM has specialised classes to handle geographic objects’ properties such as position, spatial attributes, spatial relationships, and spatial operations, and introduces two constructs: spatial aggregation, and spatial grouping. These constructs handle the construction of complex objects. A many-to-one relationship called is_located_at connects each geographic object to its position in space, and this position is represented by a positions class. An object can have more than one position if, for example, it has more than one representation (for example a city could have both a point and a polygon rep-
representation). This model uses the benefits of an OO model in representing geographic data, and as such can model relationships and operations, and complex objects (aggregations and compositions). It does not offer support for temporal dimensions.

Nevertheless, the combination of the representation of complex geographic entities as objects, and the facility to incorporate versioning mechanisms with those objects, provides a platform for an object-oriented spatio-temporal representation.

2.2.4 Temporal and spatio-temporal OO models

Temporal object models

Having considered the advantages of OO modelling on spatial and spatio-temporal data representation, the potential impact of this on OO based models will now be discussed. The OO model can be utilised directly to support time-varying information. One way is to timestamp attribute values. Consider the following type definition:

\[
\text{type employee is object}
\]

\[
\begin{align*}
\text{function name (e: employee } & \rightarrow \text{n:string}) \\
\text{function dept (e: employee } & \rightarrow f:(t:time \rightarrow d:department)) \\
\text{function projects (e: employee } & \rightarrow f:(t:time \rightarrow d:\{\text{projects}\})
\end{align*}
\]

The *name* function maps an *employee* to a particular name string. The *dept* function returns the relevant department when supplied with a time parameter. Similarly, the *projects* function returns a list of projects for an employee.

Alternatively, the employee could be timestamped at the object level, with a *get_state* operation that returns a static *snapshot_employee* object, containing non-timestamped *dept* and *projects* functions [Snodgrass (1995a)]. This example shows how simply attributes can be temporally recorded, but this is not an object model.
Temporal objects have been modelled and implemented by Sotiropoulou et al. (1998) in what they refer to as a temporal extension to the ODMG (Object Data Management Group [ODMG (2001)]) standard. This extension attempts to provide temporal data management within what it calls a temporal object-oriented database management system (TOODBMS), by implementing a temporal object data model (TODM). The model uses a common time standard based on a linear structure, where time is divided into granules, the smallest of which is called a chronon. The time entities instant, period, and interval are used, and are defined as follows:

- **instant** - a point in time, for example 28/03/06;
- **period** - the time between two instants, for example [21/03/06, 28/03/06];
- **interval** - a time period without limits, for example 1 month.

TODM handles temporal data at the object level and the instance level by subtyping attributes, relationships, and object types. A temporal object has a state, which is defined by its instance properties of *attribute* and *relationship*. An attribute is of a single type, whereas a relationship is between two types. Thus temporal objects have states that can evolve over time, and object evolutions can be recorded over both transactional and valid time dimensions.

An object can evolve over both time dimensions as a whole, and in this case its evolution is recorded at the object level. If, however, the object’s properties vary independently over one or both time dimensions, then these are defined at the specific instance level. Figure 2.3 shows temporal instance and temporal object properties. Access to the value of a temporal attribute or relationship is achieved by adding a timestamp to the object and the instance property. This model offers effective representation of evolving objects, and distinguishes time-dependent attributes that can evolve independently. The model exploits the features of association and aggregation, and thus models both object and attribute histories. No mention is made of the possibility
of modelling spatial configurations (where a feature is composed of multiple spatial objects), and the extra layer of complexity this would introduce.

Borges et al. (1999) introduces spatial data integrity constraints to assist in the OO modelling of spatial data, by extending the Object Modelling Technique (OMT [OMT (1998)]) in the form of OMT-G. This model uses geographic primitives along with the primitives of the OMT model, providing geometry and topology for geographic data, and support for topological structures, networks, multiple views of objects, and relationships. Topological and spatial relationships are also maintained as integrity constraints. The model functions at the conceptual/representation level, and has classes representing continuous, discrete, and non-spatial data. Classes can be conventional or georeferenced. Georeferenced classes are further specialised into geo-object and geo-field classes. Temporal attributes are not supported.

A model called the Temporal Versions Model, or TVM, incorporates time into a versioning system [Moro et al. (2001, 2002)]. The model is able to ver-

---

**Figure 2.3:** Temporal object and temporal instance properties (taken from Sotiropoulou et al. (1998))
version objects and store the history of dynamic property values for each versioned object. Time is associated to objects, attributes, and associations, and attributes and associations can be defined as either static (where no changes are recorded), or temporal (where all changes are recorded in a history). Two different time dimensions are used: branched time and linear time. Branched time is partially ordered: time is linear from the past to the present, where it divides into numerous timelines which can each represent a potential sequence of events or values [Snodgrass (1995a)]. This type of time requires two different instants to be successor or antecessor to another instant. Linear time is totally ordered between two instants. In TVM, branched time is used for objects representing alternate designs or versions, linear time is used for attributes and associations within each version.

Figure 2.4 shows a version hierarchy in TVM of one object. Alternative versions (also known as sibling versions) are shown as branching from the same version (for example versions V2 and V3), whereas child versions are derived from the parent object (for example version V3 is the child of parent V1). The time evolving attribute X of version V6 is shown top right. The attribute history shows the attribute value at the given valid times. Its value is 20 from 01/01/2002 on. Then, it changes to 25 on 03/15/2002,
Figure 2.5: TVM classes (taken from Moro et al. (2002))

and finally, to 35 on 05/20/2002. These values are stored as an attribute history inside the object. Other versions have the same attribute X, but with different values. Therefore, in the case of an alternative version, the same attribute may have different values for the same versioned object, represented by branched time.

The model permits two types of classes: standard and temporal versioned. Standard classes allow no versioning or temporal attributes, whereas both are supported for temporal versioned classes. Figure 2.5 shows the base class hierarchy for TVM. The TemporalObject class has methods that support temporal labels, and the TemporalVersion class contains the version attributes configuration and status. The class VersionedObjectControl has reporting functions to retrieve version information, such as the current version, the number of configurations of versions, and the first, last, and next versions.

The model also includes the version states of working, stable, consolidated, and deactivated. Versions are created in the working state, and can be modified and queried. When a version is derived, its predecessor is automatically promoted to the stable state, which allows derivation, promotion to consolidated, querying, and removal, but not modification, thereby preserving historical information. The consolidated state can be derived and
Figure 2.6: Versioned objects of the PagePattern class, showing one versioned object (Winter) (taken from Moro et al. (2002))

A version is set as deactivated at the end of its lifetime, and can then only be queried or restored to the working state.

A practical example of TVM is given by a web page that has several versions of its pages stored, which can be restored as and when required.

Figure 2.6 shows the versions of a class PagePattern, which has the versioned objects Autumn, Winter, Spring, and Summer. Winter is shown in detail. Its first version, (9,15,1), is the default page for Winter; versions (9,15,2) and (9,15,3) are derived versions and are alternative pages for Christmas (note the attribute change for version (9,15,2)); version (9,15,4) is the New Year page, and version (9,15,5) is a derivation of the new year page for the Millennium.

This model shows some of the temporal capabilities of object-orientation, such as the use of aggregation (in forming the temporal object) and the use of collection attributes to record histories. It does not go far enough for the
purposes of spatial data as composite objects (spatial configurations) are not catered for.

Object-oriented spatio-temporal models

UML has been extended to incorporate spatial and temporal attributes by Price et al. (1999) in the extended spatio-temporal UML. They maintain that the class diagram is the most appropriate to model spatio-temporal data, but that this process is not straightforward. Figure 2.7 shows two possible UML classes depicting a hospital with the time variant property *half-hour zone*.

This property denotes the area that is a half-hour travel time from the hospital. Representation (a) depicts that class with a composite attribute for the property consisting of the timestamp and spatial extent (that is geometry). Representation (b) defines the spatial property as a separate class, with an association class denoting the temporal attribute. This second approach is more complex, but is more suitable if more than one attribute is associated with a temporal or spatial extent. This model is an example of the expressiveness of an OO representation, but does not model histories or complex objects.

A concept called GeoFrame-T [da Rocha et al. (2001)] extends UML nota-
tion to include spatio-temporal data requirements. GeoFrame-T proposes a set of geospatial classes that can be used to model a temporal GIS application. It introduces the classes `GeographicPhenomenon` and `NonGeographicObject` to represent spatial and aspatial components respectively. The `GeographicPhenomenon` class gives rise to the `GeographicObject` and `GeographicField` classes which represent object and field data. Symbols are introduced to show the representation of the object being modelled, for example point or polygon. GeoFrame-T employs a `TemporalObject` class to model time in the system. The `TemporalObject` is aggregated from an `ObjectTime` and `TemporalMetadata` class. The `ObjectTime` class specifies the transaction time, the valid time, or both (bi-temporal) for the `GeographicObject`, and the `TemporalMetadata` class specifies the time reference being used (for example GMT, Universal Coordinated Time). Thus, classes defined in GeoFrame-T can vary according to the specified temporal objects modelled. This feels like a natural way to model geographic data, and the separation of time into another object maintains the integrity of geographic objects. The model attempts to reduce complexity using symbols, but the success of this is questionable. There are twenty three symbols that can be used (ten spatial and thirteen temporal), and the symbols themselves add visual complexity (see figure 2.8) and are difficult to differentiate without close study, which detracts from the inherent simplicity of the class diagram.

Figure 2.8: Simple example of a GeoFrame-T model (taken from da Rocha et al. (2001))
The Triad model [Peuquet (1994)] takes the questions “what”, “where”, and “when” as its premise, and represents data in three interrelated views: the object view (what), the location view (where), and the time view (when). The object-based view stores objects in an inheritance hierarchy, with location and time interval attributes. The time interval records the time between events, which are stored in the time-based view. The time-based view records events, their locations, and the type of change they invoke, in an ordered temporal sequence called an event vector. The locational view corresponds to a snapshot model, where overall states are stored in temporally ordered images. Thus the model represents time, location, and objects in a unified theory representing object, location, and time dimensions. The object and time-based views seem complementary, and the separation of the time dimension is a natural OO principle, as time has no direct bearing on a geographic object’s representation. The model seems to lean more naturally towards the object-based representation, but lacks any specific implementation.

MADS (Modelling of Application Data with spatio-temporal features) is a spatio-temporal conceptual model, and focuses on objects and their relationships. MADS uses ADTs to represent spatial data, and temporal ADTs (TADTs) that support timestamping. Timestamping can include the representations of instants and interval temporal elements. The model allows spatial and temporal ADTs to be represented as generalisation hierarchies. Spatiality is defined by a geometry attribute, and temporality by a status attribute. The status attribute has four states: not-yet-existing, active, suspended, and disabled. Relationship types may also be defined by the geometry and status attributes. An example of a car accident is described, where two car objects are linked by the relationship accident which is both spatial (represented by a point geometry), and temporal (time of accident).

The authors claim that to represent spatio-temporal models in UML would quickly make the diagrams complex and impractical, and so MADS uses a combination of shapes and symbols to simplify the schema. Figure 2.9
Figure 2.9: MADS schema diagram (taken from Parent et al. (1999))
shows an example MADS schema for part of a river monitoring application. It is argued that the diagram has gone some way in overcoming the inherent complexity introduced by the time dimensions, but has veered unnecessarily far away from UML. For example the aggregation/composition associations could easily have been kept to standard UML notation, with topology symbols placed along the edge and cardinalities at each end. The MADS system has been implemented as a visual editor in Java, allowing diagrams to be created and manipulated via a graphical user interface [Parent et al. (1999)].

All these models attempt to balance the expressiveness of their representation against complexity. At the highest level of abstraction is the Triad model, which attempts to unify the concepts of objects, fields, and time. The model successfully integrates all three views, but is at too high a level to form the basis of a practical representation. MADS and GeoFrame-T provide powerful modelling of spatio-temporal data, but their attempts to reduce complexity in their diagrams are only partially successful. GeoFrame-T extends UML but the addition of its own symbols makes the diagrams harder to comprehend, and MADS diagrams rely more heavily on extra, specialised symbols and therefore exhibit the same trait. The OMT-G model strays less far from the standard OMT methods, and utilises the benefits of OO, but does not include time. However, it is already complex, although powerful, and this complexity would be deepened by the inclusion of temporal attributes. TODM also exploits OO technique, and uses these effectively to model object evolution over time, and represents attribute histories over both valid and transaction times. These are notable techniques in the representation of object evolution, and one can see that this has potential for spatio-temporal data if spatial attributes were included. However, its bi-temporal nature increases the complexity of the model, and might prove problematical in an implementation during querying. The two models that adhere closest to UML conventions, extended spatio-temporal UML and TVM, both minimise the impact of time by restricting its inclusion to valid time only, or
valid and branching time, respectively. Like TODM, TVM represents attribute histories as a collection of timestamped attributes in linear time, and the model uses branched time for objects, which allows different versions of the same object to exist in parallel. The model does not apply to geographic data, though, and does not attempt to represent complex objects. Extended spatio-temporal UML includes a valid time attribute and spatial extent, and its representation thus manages to keep the diagram simple. However, complex objects and histories are not catered for.

2.3 Representing change and events in spatial data

2.3.1 Features and change

The addition of time into a SDBMS and GIS introduces further complexity, and it is useful to draw comparisons between the spatial and time dimensions. Table 2.2 shows such a comparison (taken from Langran and Chrisman (1988)). By considering these differences, we can isolate temporal topology from spatial topology. So, where spatial queries might ask of an object “what are its neighbours?,” “what are its boundaries?”, or “what encloses it?”, temporal queries might ask “what was the previous state?”, “what has changed over a given time?”, or “what trends are evident?” [Langran and Chrisman (1988)].

Types of change that features can undergo are:

- interior change: cohesiveness, internal structure
- boundary change: geometry, permeability, fuzziness
- creation, modification, deletion
- introduction to and elimination from the hierarchy
<table>
<thead>
<tr>
<th></th>
<th>Space</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>map</td>
<td>state</td>
</tr>
<tr>
<td>Meaningful units</td>
<td>objects</td>
<td>versions</td>
</tr>
<tr>
<td>Object separation</td>
<td>boundaries</td>
<td>mutations</td>
</tr>
<tr>
<td>Measurement</td>
<td>length, area</td>
<td>duration</td>
</tr>
<tr>
<td>Position</td>
<td>coordinates</td>
<td>dates, times</td>
</tr>
<tr>
<td>Contiguous neighbours</td>
<td>adjacent objects</td>
<td>previous/next states</td>
</tr>
<tr>
<td>Max contiguous neighbours</td>
<td>infinite</td>
<td>two</td>
</tr>
</tbody>
</table>

Table 2.2: Comparison between spatial and time dimensions

- movement between levels of the hierarchy
- merging and splitting of levels
- changes of linkages between levels [Worboys (1998)].

One important aspect of time management is time granularity, for example days, weeks, months, years. Some features will vary according to seasons (for example forest plantations), whilst some will vary on a daily or hourly basis. For this reason, time in GIS has been studied mainly in two areas, that of moving objects, and that of evolving objects. As mentioned earlier, the term ‘moving objects’ implies a shorter time granularity than that of evolving objects, and is outside the scope of this research.

### 2.3.2 Event-based spatio-temporal models

As stated in the previous chapter, a comprehensive treatment of change must consider events and processes, and examples of models that provide this will now be discussed.

A model for representing dynamic processes is described by Worboys (2005). In this model, the progression of time is represented by a sequence of processes called ticks. Each tick can be connected to an entity via a channel called a tocc. A particular sequence of ticks forms a clock. Space is divided
into a subdivisions called $loc$s, and each $loc$ can be connected to an entity via a channel called a $socc$. In this way, an entity can be connected to a time and a location, and its movement modelled. The example provided is that of the motion of a vehicle, tracking its location at different times using equations. The main area of development in this model is the application of calculi to real-world occurrents, and is more suited to the modelling of moving objects.

The geospatial event model (GEM) [Worboys and Hornsby (2004)] constructs a representation of dynamic geospatial domains using object representations for spatial entities and events. The GEM model uses settings as a characteristic of a spatial entity. A spatial object is situated in a setting. Settings can be spatial, temporal, or spatio-temporal in nature. These settings are abstracted into the classes $SpatialSettingClass$, $TemporalSettingClass$, and $STSettingClass$ respectively. Spatial objects and event objects are mapped to a setting using a function called $situate$. Figure 2.10 shows the relationships between objects, events, and settings in the GEM model. The model consists of the following:

- Spatial object classes, instances, and object-object relationships.
- Event classes, instances, and event-event relationships.
- Settings in which spatial objects and event objects are situated.
- The situation function that maps spatial objects and events to their settings.
- Spatial object-event, and event-object relationships.


- Initiation - an event A starts an event B. For example, a traffic light turning green initiates the forward movement of a vehicle along a road.
Figure 2.10: The GEM model showing objects, events, and their interaction (taken from Worboys and Hornsby (2004))
• Perpetuation/facilitation - event A plays a positive role in initiating or sustaining event B. For example, opening a second toll booth facilitates traffic flow over a bridge during rush hour.

• Hindrance - event A plays a positive role in weakening or stopping event B. For example, the closing of a toll booth slowing down traffic passing over a bridge.

• Termination - event A forces the termination of event B. For example, running out of fuel terminates the progress of a vehicle.


• Creation - an event that creates a new object, for example, a bridge building event creates a new bridge.

• Sustaining of being - an event that results in the continued existence of an objects, for example, the painting of a bridge may result in its existence being extended.

• Reinforcement/degradation - an event that has a positive/negative effect on an object. For example, clearing snow of a road keeps it open, whereas a storm event might decrease the functionality of a bridge.

• Destruction - an event that results in the destruction of an object. For example, and explosion may destroy a building.

• Splitting/merger - and event that creates/destroys a boundary between objects. For example, the boundary between East and West Germany.

The GEM model uses object-orientation to effectively represent spatial objects and events, and relationships between them. However, it does not manage histories, so an object’s spatial configuration could not be tracked through time, nor could its evolution be represented by a sequence of events.
The example scenario used to apply the model, that of an airport and associated events, suggests the model is more applicable to moving objects. It also requires an extended, non-standard version of UML to represent as a class diagram.

Another model that uses classes for objects and events is described by Vidal and Rodríguez (2005). This representation uses C-logic [Chen and Warren (1989) cited in Vidal and Rodríguez (2005)] and event calculus to specify its model. Spatial properties that apply to objects and events are described by attributes. A spatial object is an instance of a class and is time-varying. For example, one class instance representing a building could be a school at one time, another instance of the same class could be a hospital at another. Relationships between objects are specified by events. Event attributes, however, hold for the specific time in which the event occurs and do not change with time. In common with the GEM model, events for creation, destruction, splitting, and merger are represented, as well as a more general change event, which can apply to both spatial and aspatial properties of an object. Changes between spatial objects are specified in terms of event occurrences, and the model can also represent event-event relationships. There is no diagrammatic representation provided, and processes and histories are not supported.

Chen and Jiang (2000) present an event-based model to represent spatio-temporal data that describes change in land subdivision systems. The model represents land parcel data and events associated with changes to their boundaries, such as site location, allocation of land use permits, and title registration. The model is relational in nature, and stores event data in table rows (tuples) with an identifier. These tuples describe the relations between spatial objects and related events. The events modelled can be comprised of other events, some of which are conditional to other events. For example, permits cannot be issued until site location has been established. The sequence of these events make up a workflow for the system and generate
new spatial objects. The model uses mathematical definitions in the form of event expressions that define event-event and event-object relationships, to ensure that event conditions are not violated. An event expression consists of primitive events and event operators. So, for example, the composite event ‘permit’ could only be completed by the completion of an ‘issue permit’ event OR a ‘notify rejection’ event. This example uses the OR operator. Other operators can be used to formulate event expressions, such as AND, ANY, and NOT. These expressions are Boolean, and event time is represented as a time instant value. This model can be considered a chronicle-based model in the sense that it tracks the change to land parcel boundaries by recording a sequence of events. The model’s drawbacks are that it is relational, and therefore will be subject to the limitations this model imposes when representing complex objects. Indeed, the authors acknowledge that the implementation cannot represent complex events, and that these must be derived by functions based on event primitives.

Ikazaki and Watanabe (2006) describe a model that can represent geographic change as events. The model uses three relations in its representation: the is-a relation, the part-of relation, and the event-driven relation. Alongside these event relations is the spatial concept and the spatio-temporal object. A spatial concept is described as any object that can be identified in geographic space, and the spatio-temporal object represents an ‘invariable state’ of a spatial concept. Spatio-temporal objects are associated with events (see figure 2.11), and have attributes for the onset time and the termination time of the event. The is-a event relation enables events to be treated from different viewpoints. For example, a typhoon event could be considered as a disaster event or as a storm event. The part-of relation enables events to be themselves parts of other, composite events. In this way, a storm event could be constructed by aggregating multiple downpour event primitives. The event-driven relationship represents causal relationships between events. For example, a tsunami event could cause a disaster event. This model
Figure 2.11: Overview of the model described by Ikazaki and Watanabe (2006)

Figure 2.12: Evolution of a river boundary (taken from Galton (2004))
provides a flexible and descriptive mechanism for describing events and how they induce change in geographic features. Although it describes how composite events are modelled, it does not explicitly represent complex change in composite, spatio-temporal objects. It also does not demonstrate how a succession of events and associated spatio-temporal objects are represented.

Galton (2004) describes a set of data structures that could be used to represent an object-based view of events. The structures represent a sequence of events related to the evolution of a river boundary, as depicted in figure 2.12. In this scenario, a river $R$ separates two countries, $A$ and $B$. These countries share a border denoted by the course of the river. At time $t0$, a treaty is agreed that the border should remain coincident with the river, as long as changes to the river’s course are gradual. It is also stipulated in this treaty that if a sudden change occurs to the river’s course, then the section of the border coincident with the part of river that undergoes this sudden change should remain in its new position. Between times $t0$ and $t1$, the river undergoes gradual change, but at time $t1$, it undergoes a sudden change when a meander is broken through creating an ox-bow lake. In accordance with treaty, the land enclosed between the ox-bow lake and the river remains part of $A$, even though it is now situated on $B$’s side of the river. At time $t2$, a new treaty is agreed, stipulating that the border is once again coincident with the course of the river, and transferring ownership of the ox-bow lake and enclosed land to $B$.

This scenario can be represented by the data structures shown in tables 2.3 and 2.4. For continuants, shown in table 2.3, the type, ID, and associated events are specified. For events, shown in table 2.4, its type (punctual or durative), ID, time, and initial and final states are specified.

In this model, the positions of continuants is not given directly in the continuant data structures, as a continuant’s position is not seen as fixed. Rather, positions are related to continuants via the events that caused them to change. In the example, the event for the meander cut-off is represented as
punctual, and has a single time allocated to it \((t_1)\), but refers to initial and final states, which is an ambiguity - which position holds for time \(t_1\)? This emphasises a difficulty when representing events in static data structures: clearly the meander cut-off would not occur instantaneously, but the known, or recorded, time for the cut-off may only be a single value. This is explained by Galton by the statement “in reality we usually have to be content with a state of partial ignorance”. Nevertheless, this model shows that recording events in the life of a continuant can be used to record its life history using an OO approach, and is an example of the chronicle-based view mentioned in section 1.2.2.

The spatio-temporal data model (STDM) described by Wachowicz (1999), however, uses an OO approach to model changes as they relate to objects in a state - event - state approach. The model uses space-time paths, which are a combination of states of objects linked by events. The design of the model is based on the following principles:

- The model emphasises the interaction between events and states.
- The space-time path offers a semantic abstraction that handles only valid time.
- A versioning mechanism is required to represent change in a database system, and that extensions to the relational model are not adequate
EVENT   Type: Punctual
ID: MCO (meander Cut-Off)
Time: t1
Initial: Position of R is P1
Final: Position of R is P'1

EVENT   Type: Punctual
ID: Treaty1
Time: t0
Initial: -
Final: AB coincides spatially with R so long as the position of R changes continuously

EVENT   Type: Punctual
ID: Treaty2
Time: t2
Initial: -
Final: AB coincides spatially with R so long as the position of R changes continuously

EVENT   Type: Durative
ID: RCD1 (River-Course Development)
Time: (t0, t1)
Initial: Position of R is P0
Final: Position of R is P1

EVENT   Type: Durative
ID: RCD2
Time: (t1, t2)
Initial: Position of R is P'1
Final: Position of R is P'2

Table 2.4: Events in an object-based event model (taken from Galton (2004))
Inheritance is used to represent incremental modification to a state in the space-time path. Modification occurs as a result of an event. Events are represented as instances of classes, and hold a timestamp value, which can be an instant or an interval type. Thus, the space-time path for a geospatial object is constructed by connecting each instance of a class with a corresponding instance of another class, this connection depending on the modification mechanism involved.

The model was evaluated by applying it to the scenario of evolving public boundaries. These boundaries can have six possible states: draft, proposed, new, disputed, old, and obsolete. The public boundary can evolve through these states via events. These events are allocation, delimitation, demarcation, and administration. So, for example, a draft boundary can become a new boundary as a result of a delimitation event. In this case, the event has two timestamps representing the operative date and the effective date of the boundary. The operative date is the date at which the boundary was issued by an act or order of parliament, and effective date is the date at which the boundary became effectual.

This model is a natural way of modelling feature evolution, but does not explicitly represent individual changes to objects of arbitrary complexity.

A model that uses identity to track change in an object is described by Hornsby and Egenhofer (2000). (This is a more detailed description of the brief outline of a method for representing object mutation, described in the previous chapter.) This model tracks an entity's evolution with a history graph and defines four primitives (see figure 2.13):

- **object existence** - denoting that an object is present

- **non-existence with history** - an object has existed, but has been eliminated and no longer exists
(a) object existence  (b) non-existence with history  (c) non-existence without history  (d) transition

Figure 2.13: History graph primitives (taken from Hornsby and Egenhofer (2000))

- **non-existence without history** - the object does not exist, and has never existed
- **transition** - shows the change from one state of an object to another []

From a data perspective, non-existence with history would mean that an object that has existed in the database but has been deleted during an update would retain records of its previous existence. This would be of considerable benefit in, for example, a system storing road network data, as a road configuration that has been altered due to a section of road being destroyed could be reconfigured. Non-existence without history would mean that the object representing the section of road would be deleted along with its previous states, meaning that the system has no record of its being.

Using these primitives, nine possible transitions can be identified, as shown in figure 2.14.

The model then introduces a cross transition between objects, where one object induces change in another. This extends the possible transitions to eighty one. Using these transitions, it should be possible to model the evolution of change in an entity by sequencing transitions. This model provides a system of representing change, which could be useful when conceptualising a data model for a GIS at a high level, but time is not explicit in the graph, and the representation of change does not seem natural. To take our forestation example, if a forest object were to be altered by disease, then this would be
(a) continue non-existence without history (b) create (c) recall (d) destroy (e) continue (f) eliminate (g) forget (h) reincarnate (i) continue non-existence with history

Figure 2.14: History graph transitions (taken from Hornsby and Egenhofer (2000))
represented by the destruction of the object and its reincarnation in the new form. The old object can still exist in the model, with a history if necessary. This model provides a visual representation of the concept of object change, and the order of states as they undergo transition, and concurrent states, are represented. However, quantitative measures of time are not provided, meaning that the actual time or duration of transitions is not represented.

The Event-based Spatio-temporal Data Model (ESTDM) [Peuquet and Duan (1995)] represents a set of changes at grid-based locations for a geographic theme (event), and organises these changes in a temporal axis in the form of an event list. Even though this model is event-driven and time-based, it is location-based rather than feature-based, and is therefore appropriate to raster data.

These event models can be broadly divided into two groups: those representing events as objects, and those that represent events as mathematical functions. The exception is the GEM model, which does represent events as objects, but establishes relationships between events, spatial objects, and their settings using a \textit{situate} function. The model described by Worboys (2005) models events at a lower level, matching a sequence of dynamic processes and relating them to locations using event calculus. The examples used to illustrate the models in both cases suggest that these models are more relevant to modelling the dynamic processes involved in moving objects. The models described by Chen and Jiang (2000) and Vidal and Rodriguez (2005) similarly represent relationships between events and spatial objects, and between events and other events, with mathematical functions. Vidal and Rodriguez (2005) define classes for spatial objects and events, although not diagrammatically, but Chen and Jiang’s model represents these as relational tuples, and does not provide an entity-relationship diagram to clarify the model. Neither model provides an explicit representation of history, either through linking successive spatio-temporal objects, or linking successive events. The models described by Galton (2004), Ikazaki and Watanabe
(2006), and STDM [Wachowicz (1999)] represent events at a higher level without the use of functions. Ikezaki and Watanabe’s model is the more expressive in its representation of compound events, and is similar to Galton’s in that it represents spatial objects as incidental to events, the event being composed of event-generated spatio-temporal objects. Galton’s model is not really a model, but more of a description of how simple data structures can provide a representation of event-based spatio-temporal data. As with STDM, timestamps are applied to events, and not the spatial objects, due to the objects position not being fixed in time. This is in keeping with a truly event-based view, where states of spatial objects are incidental to the events that modify them. Only STDM represents histories explicitly, by a version graph created by the inheritance hierarchy of object versions, although Galton does suggest that an event list, if constructed, could represent a life-history of a spatial object. Neither Galton’s model or STDM represent compound events. The model of identity-based change [Hornsby and Egenhofer (2000)] is powerful in describing complex change in spatial objects, and its concepts can be Incorporated into a data model, as Galton (2004) demonstrates.

2.4 Implementations of STDBMS

Commercial GIS with spatio-temporal capabilities

Two commercial GIS offer support for versioning geographic data. These are Smallworld GIS [Easterfield et al. (1990)], and ArcGIS [ESRI (2006)]. Both employ versioning of spatial data aimed at simultaneous updating and modification by multiple users, and reconcile different versions into a unified state. Smallworld’s model is object-oriented, and events can be included. ArcGIS uses an essentially relational data model, and uses database states to store historical stages of development.
Tripod

Tripod [Griffiths et al. (2001)] is an object-oriented database management system (OODBMS) built from scratch that can manage spatial objects that change over time. Tripod allows the database designer to define his/her own data types (also known as user defined types or UDTs). Object evolution is supported through the maintenance of a history, which records changes made to an object’s state, attributes, or relationships by identifying these changes with a timestamp. It achieves this by extending the ODMG primitive types with two temporal literals, called \textit{Instants} and \textit{TimeIntervals}. These are described as one-dimensional extensions (time) of a two-dimensional realm (spatial). These types exist in a “temporal realm”, which is implemented as a finite set of integers. \textit{Instants} are a collection of timestamps, and \textit{TimeIntervals} are a collection of pairs of timestamps, where the first is the start and the second the end, of a contiguous time-interval. A Tripod timestamp, therefore, is either an \textit{Instants} or a \textit{TimeIntervals} type. These timestamps allow histories to be defined and queried.

The system maintains the set of states in a \textit{Histories} object. This stores a set of pairs for an object, where one is a timestamp, and the other a snapshot of the object.

Tripod supports spatial, temporal, and historical queries by providing an extension to the object query language (OQL [ODMG (2001)] called Tripod-OQL. Examples of these are now given based on a land-use application:

- Temporal query: \textit{Which counties were founded before Avon?}

- Spatio-historical query: \textit{What are the land parcels that at some point in time bordered parcel 2601?}

- Historical query: \textit{Display the previous versions of parcel 2604’s geometry.}

- Spatio-historical join: \textit{What are the neighbouring parcels of parcel 2586}
Tripod provides a binding to C++ to support application development [Griffiths et al. (2001)].

**Temporal GIS in ArcInfo**

A prototype temporal GIS (TGIS) is described by [Candy (1995)]. The implementation is an extension to ArcInfo, and is based on what is described as the vector update model, but is in fact much more akin to the ‘base state with amendments’ [Langran (1992)] model, as only the current state is stored in a layer, and vector updates are stored in separate layers. The implementation suffers from the inherent drawback of the model, that is complex overlay operations to retrieve historical states, and does not maintain topology for feature versions. The implementation is now out of date in that it cannot be used with recent versions of ArcInfo, but does implement successfully some TGIS functions, notably temporal data set creation, editing, display, and query.

**Changes to administrative boundaries**

There have been some implementations of systems in Europe to record evolutionary change in administrative boundaries [Gregory (2002)]. The simplest system uses what is called the key dates approach (similar to time-slicing), where the boundaries of administrative units are digitised at particular dates, and linked to associated socio-economic data. This is a very simple and cost-effective method, but suffers from the inherent problems with time-slicing, and also an update problem if boundary changes occur between key dates. This method has been used to create an historical record of Prussia [Winnige (2000)].

Another system developed in Belgium extends the key dates idea. Here Belgium was split into nine provinces and the boundaries digitised with each
province stored in a separate file. At each boundary change, a new file is generated, called a concordance file, which is used to identify which boundaries belong with which data. In this way a map of Belgium at a particular date can be generated by passing the date to the concordance file to retrieve the province boundaries.

A system described by Ott and Świaczny (1998) was proposed to represent boundary change in Palatinate, Germany, based on the space-time composite model [Langran (1992)]. The proposed system is constructed as follows: firstly a database is created containing the attribute data for each boundary, then a layer of digitised boundaries is created, forming the base layer. When a boundary change occurs, a new layer reflecting the new boundary is created, and overlaid with the existing one, forming polygons denoting the differences between boundaries. These polygons are given ids and linked to the boundary that caused the change. To recreate a layer at a certain date the ids are retrieved along with the base state polygons. The benefits of this system is that it represents changes over time, enabling such queries as “how did this unit change over time?”, but the disadvantage is that it requires much more GIS expertise to implement than the previous methods.

Monitoring land use change

A TGIS extension to the commercial GIS ArcInfo which implements Langran’s space-time composite model is described by [Raza et al. (1996)]. The application is aimed specifically at the representation of urban land use change. The underlying data model is relational, and the system uses tuple-level versioning to implement the model. The model is based on the spatio-temporal-attribute object (STAO) model, as shown in figure 2.15, which encapsulates three components of real world entities: location (spatial), attribute (aspatial), and time (temporal). The STAO can itself be decomposed into a spatio-temporal object (STO) and an attribute-temporal object (ATO). The model distinguishes attributes as either essential or non-essential
Figure 2.15: STAO structure (taken from Raza et al. (1996))

(similar to design attributes in CAD object versioning), where a change in an essential attribute creates a new STAO, whereas a change to a non-essential attribute creates a new version of the existing STAO. Change is classified at three layers: object, properties, and change. The object layer has object and version components, the properties layer has essential and non-essential components, and the change layer has essential and non-essential components. Change is visualised by generated overlays. The application was tested using data from Dar-es-Salaam, Tanzania. Figure 2.16 shows the results from the following queries:

Query: Display all LU ‘unplanned residential high density’ in 1982
Outcome: Result A

Query: Display all LU ‘unplanned residential high density’ in 1992
Outcome: Result B
Figure 2.16: Query results from the STAO test case (taken from [Raza et al. (1996)]). LU stands for Land Use.
Query: Display all LU which changed to ‘unplanned residential high density’ in 1992 from 1982 or earlier

Outcome: Result C

Query: Display all LU which changed from ‘unplanned residential high density’ to others in 1992 and later

Outcome: Result D

The results from these queries demonstrate the strengths and weaknesses of the underlying space-time composite model. On the one hand, change over time is depicted, and there is no reason why successive changes could not be represented on one map with an attribute key. However, it is argued that depicting successive changes would show how complex a space-time composite can become, thereby losing clarity. Further, although the STAO model is object-oriented in its representation, its implementation is relational, and built on existing extensions to ArcGIS. There is no explicit mapping of the OO structure of the model to the underlying data structures of the implementation (by way of an ER diagram, for example), meaning that the model is specific in its implementation platform, and that the benefits of an OO design (as outline in section 2.2.3) are not apparent.

Summary

This chapter has described the fundamental principles of geographic data modelling and some of the techniques used to incorporate time into spatial data. It is clear from the descriptions of commercial STDBMS that a relational implementation is not adequate to fully represent spatial data, and that an OR approach is required as a minimum. Even so, current STDBMS offer spatial data types without exploiting the full potential of the OR model,
and this contention is further highlighted by attempts at extending the relational model, such as STRM and GISER, which cannot overcome the limitations of ER modelling when applied to spatio-temporal data, specifically, the complexity inherent in the data. The ER approach does not have sufficient powers of abstraction necessary to represent such data effectively, as shown by the attempts of the modelling units [Tryfona and Jensen (2000)] and GISER [Shekhar et al. (1997)] models described in section 2.2.2. The ER-based models also lack the potential benefits offered by an OO approach, as described in section 2.2.3. The discussion on OO modelling emphasises the power and expressiveness made available by these benefits. However, despite this, there remains a significant obstacle to successful spatio-temporal representation: complexity. The special requirements of spatio-temporal data mean that attempts to modify OO modeling techniques, such as the MADS model, have not overcome this complexity but merely altered its representation. Further, these models do not explicitly represent events, and these must be catered for in a spatio-temporal model. The inclusion of events into the model introduces further complexity due to the interaction between events and spatial objects, and between events and other events. The representation of these relationships requires further abstraction capabilities, and has resulted in the use of mathematics is some of the models described in section 2.3.2 to describe these relationships. Other models, however, [Ikazaki and Watanabe (2006); Galton (2004); Wachowicz (1999)] have been able to utilise the abstraction capabilities of OO to represent events within a spatio-temporal model. Nevertheless, none of the event-based models described explicitly represent spatio-temporal data that includes complex spatial objects, compound events, and histories of either spatial objects or events. It is argued, therefore, that to include these structures it is necessary to fine-tune the requirements of a spatio-temporal data model, to make a STDBMS or temporal GIS sufficiently powerful and practical in its implementation. This means that a STDBMS cannot be ‘all things to all people’ and must make
compromises in its structure to satisfy the requirements of its application domain.

In the next chapter, these principles will be expanded upon and expressed in the FEM spatio-temporal data model.
Chapter 3

The FEM spatio-temporal data model

This chapter will describe the FEM model. Section 3.1 will describe the foundation of the model and provide a high-level description of the model’s types and the associations that provide the spatial and temporal structure. Section 3.2 will describe the model’s core types, their attributes, and the basis for their structure. This will include the object versioning techniques that have been employed, made possible by utilising an OR approach, and how events are represented. It will be shown that through these techniques, an absolute-hybrid spatio-temporal data model can be described with just five types, providing the basis for the implementation of a temporal GIS. Section 3.3 will demonstrate how the FEM model copes with some real-world data. (This research is funded by an EPSRC CASE award and is carried out in collaboration with the Ordnance Survey, and as such uses Oracle Spatial, and Ordnance Survey Master Map Integrated Transport Network data, henceforth referred to as ITN data).

The terminology used to describe geographic data within a temporal context conforms to ISO standards specified in ISO/TC 211 Geographic information/Geomatics (2000).
3.1 The foundation of the FEM data model

It is the contention of this research that OO methods offer a more structured and expressive means of spatio-temporal representation, and that OO techniques can be utilised in an absolute-hybrid model. These methods allow geographic objects and time to be better abstracted as states, with events modelled with equal status. A collection of timestamped states can be grouped together to produce a single compound spatio-temporal object whose values represent successive states across time. Similarly, events can be modelled as distinct timestamped objects, and reference, and be referenced by, states. Further, the representation of events as objects means that they can also be aggregated into compound objects to form a chronicle-based view of the changes to complex geographic objects. These principles are at the core of the FEM representation.

3.1.1 Components of an optimal data model

Before describing the FEM conceptual data model, it is worth considering what elements are required for a data model, and what criteria are necessary to optimise the model.

A data model can have the following components [Connolly and Begg (2002)]:

1. a structural component, consisting of elements from which a database system can be constructed

2. a manipulative component, defining operations on data, including those for managing the data by way of updates, inserts, and delete operations

3. a set of constraints to ensure data integrity.

For the FEM conceptual data model, the structural component is in the form of type definitions. The manipulative component is comprised of SQL queries
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural validity</td>
<td>Consistency with the way the enterprise defines and organises information</td>
</tr>
<tr>
<td>Simplicity</td>
<td>Ease of understanding by Information Systems and non-technical users</td>
</tr>
<tr>
<td>Expressibility</td>
<td>Ability to distinguish between different data, relationships between data, and constraints</td>
</tr>
<tr>
<td>Non-redundancy</td>
<td>Exclusion of extraneous information</td>
</tr>
<tr>
<td>Shareability</td>
<td>Not specific to any application or technology and thereby usable by many</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Ability to evolve to support new requirements</td>
</tr>
<tr>
<td>Integrity</td>
<td>Consistency with the way the enterprise uses and manages information</td>
</tr>
<tr>
<td>Diagrammatic expression</td>
<td>Able to represent the model using an easily understood diagrammatic notation</td>
</tr>
</tbody>
</table>

Table 3.1: Criteria for an optimal data model (taken from Connolly and Begg (2002))

and spatial operations, and these elements will be described in chapters 5 and 6. The constraints of the model are those imposed by the data structures in the relationships between objects, and the need to track the identity of objects as they evolve. These will be detailed in the ensuing type definitions.

Table 3.1 details the criteria for an optimal data model. After detailing the model, its structure will be evaluated against these criteria.

3.1.2 Basis of the design

At the heart of the design of a spatio-temporal model is the question of which data structures to use. In this case, UML classes will be used to describe a set of database objects to represent evolving geographic features. The model
builds on the versioning techniques employed by TVM [Moro et al. (2002)], and the simple spatio-temporal representation offered by extended spatio-temporal UML [Price et al. (1999)]. Objects are versioned at the object and feature level and are referred to as states. The versioning principles used are based on those of Chou and Kim (1988), where what is called a ‘generic version’ maintains a history graph of all versions of that object. A similar technique is employed by the FEM model, which uses a separate class to maintain a history of all states of a feature. Unlike Chou and Kim’s model, the FEM model does not use Dynamic Reference Resolution as this mechanism is more suited to the requirements of the engineering design environment (see section 2.3.2). The representation of change is achieved by drawing on the ideas of identity-based change described by Hornsby and Egenhofer (2000), and the concept of continuants and occurrents given by Worboys (2005), adjusted to enable the representation of complex spatio-temporal objects. In this way the model captures events as objects with equal status to spatial objects and features. Due to the potential complexities created by the introduction of time, as evident in the TODM, GeoFrame-T, and MADS models, only valid time is used to timestamp spatial objects and features (as is the case with STDM [Wachowicz (1999)]. Other aspects of the model are as follows:

- feature identity is tracked by a topographic object identifier (TOID)
- the time associated with each change increases linearly from an existing base state, this state being either the first entered, or earliest representation in the database system (change to a feature’s component spatial objects may be asynchronous)
- state is defined as a condition that exists for a period in time
- an object’s attributes, both spatial and aspatial, describe its state at a particular time, and these attributes change as a result of a particular event transforming it to a new state
the event object links spatial_object_states by referencing both an initial state, that is the state prior to the event, and the modified state, that is the state created by the event, and possesses attributes relating to the nature of the change.

geographic objects are aggregated to form features, and similarly, events between object states are aggregated to form transitions.

transitions link feature_states by referencing an initial and an evolved state.

Figure 3.1 shows a class diagram of the core FEM types and their associations. These core types have been designed with the aim of representing ITN data, which will be described in detail in the next section. The spatial_object_state object represents the invariant attributes of a spatial object. The spatial attribute of this object is a geometric primitive (that is an object representing a single, connected, homogeneous element of geometry) and is determined by the application domain and specified by subtyping. Spatial_object_states reference an event object, which describes the event which created that state. Spatial_object_states aggregate into feature_states that represent a spatial configuration for that feature. Feature_states reference a transition object, which is an aggregate of the events that describe the changes to the spatial_object_states that have varied between this feature_state and the previous feature_state. In this way, the transition object provides a chronicle-based view of the feature’s evolution. The evolved_feature provides a container for the entire recorded history of feature_states, thus providing a configuration-based view.

The model can be viewed as a three tier representation. In the bottom tier, the geometric primitives exist as a heterogeneous layer of spatial_object_states linked by events. At the middle tier, geometric primitives are aggregated into feature_states (configurations), and events are aggregated into the chronicle attributes of transitions, which link feature_states.
Figure 3.1: Class diagram of the ITN FEM data model

- FEATURE_ID : VARCHAR2
- INITIAL_TIME : DATE
- CURRENT_TIME : DATE
- STATE_COUNT : NUMBER
- DEFAULT_STATE : NUMBER
- NEXT_STATE_NO : NUMBER
- EVOLUTION_PATH : COLLECTION OF REF OF FEATURE_STATE

1

FEATURE_STATE

- TOID : VARCHAR2
- FEATURE_NAME : VARCHAR2
- VALID_TIME : DATE
- STATE_NO : NUMBER
- EVOLVER : REF OF TRANSITION
- ELEMENTS : COLLECTION OF REF OF SPATIAL_OBJECT_STATE
- MBR : SDO_GEOMETRY

0..1

TRANSITION

- TRANSITION_ID : NUMBER
- TOID : VARCHAR2
- CHRONICLE : COLLECTION OF REF OF EVENT
- INITIAL_STATE : REF OF FEATURE_STATE
- EVOLVED_STATE : REF OF FEATURE_STATE

0..1

SPATIAL_OBJECT_STATE

- TOID : VARCHAR2
- VALID_TIME : DATE
- STATE_NO : NUMBER
- MODIFIER : REF OF EVENT

0..1

EVENT

- EVENT_ID : NUMBER
- TOID : VARCHAR2
- NAME : VARCHAR2
- TYPE : VARCHAR2
- INITIAL_STATE : COLLECTION OF REF OF SPATIAL_OBJECT_STATE
- FINAL_STATE : COLLECTION OF REF OF SPATIAL_OBJECT_STATE

1..*
At the highest level, the evolved feature containers maintain a complete history of all recorded feature states for each feature.

Detailed descriptions of the FEM types will be given in the following section.

3.2 FEM type descriptions

3.2.1 The spatial_object_state type

This type represents the geometric primitives that compose a feature, and has a valid_time attribute which, for ITN data, is the “time at which the data was first entered into the Ordnance Survey database” [Ordnance Survey (2006)]. For the purposes of the FEM model, the attribute will represent the version date for the data representing this object, that is, the time when this version of the data first became available. This is a single value and is the only type of temporal information provided in the ITN data, and so no inferences can be made as to the lifespan of the object. It is also the reason that this attribute is represented as a time instant and not an interval data type.

The state_no attribute denotes its position in the state hierarchy. The created_by attribute references the event object that mutated it from its previous state (this will be null in the case of a base state object). This type represents the non-varying attributes of the spatial object - all spatial objects will possess these attributes, and the variant attributes, that include the geometric primitive, are specified by subtyping. This can be considered a form of type versioning, as used by Ahmed and Navathe (1991) in CAD database systems. Spatial_object_states and their subtypes are uniquely identified by the combination of their state_no and TOID attributes.
3.2.2 The feature_state type

The feature_state type is an aggregate of its constituent spatial_object_states, and represents a spatial configuration for the feature_state at a given time, denoted by its valid_time attribute. As with the spatial_object_state type, this attribute represents the version date for the data comprising this feature, including the configuration of spatial_object_states of this feature represented in the elements attribute. This configuration will be comprised of spatial_object_states whose valid_time attributes may be equal to, or earlier than, the valid_time attribute for the feature_state. This is because a feature_state comprised of, for example, five spatial_object_states, may have had changes to only two of these since its previous state, meaning that the elements attribute will be comprised of references to the two changed spatial_object_states, and the three unaltered ones, which retain their earlier valid_time attributes.

The identity of the feature_state is a unique combination of its TOID and state_no attributes. The type has two associations: the evolver attribute references a transition object which records the changes in the feature_state's constituent spatial_object_states from its previous state. This may involve changes to only some of the spatial_objects which comprise the state's configuration. The elements attribute is a collection of references to the feature_state's constituent spatial_object_states, which represents its configuration. One other spatial attribute included is the BR (minimum bounding rectangle), which is a rectangular spatial object that denotes the bounding box of the feature_state. This is necessary for application requirements, which will be described in chapter 6.

3.2.3 The event type

Before the event type is described, some detail about the nature of events in INT data will be given, and the implications this has on their representation.
The events that can be represented are based on the types of change described by Worboys and Hornsby (2004) and Vidal and Rodriguez (2005) (as described in 2.3.2). Typical events that can be represented are as follows:

- Creation - an event that creates a new object.
- Destruction - an event that results in the destruction of an object.
- Splitting/merger - an event that creates/destroys a boundary between objects.
- Change - an event that changes the spatial property of an object.

More detailed descriptions of how events are represented by the FEM model with INT data will be given in sections 3.3.3. and 4.4.

The representation of these events is restricted by the valid_time values for feature_states and spatial_object_states. As mentioned previously, for INT data these values represent version dates for the data, and this has the following implications for representing events:

- The valid_time values are instants, and therefore provide no information as to the duration of the state’s existence. Further, as the dates apply to data, it is not known at which time of the object’s lifespan the data relates to (a state of partial ignorance).

- Given that the valid_time values are instants and cannot be taken to denote with any certainty that the objects they apply to were valid at that time, or to what time in the objects lifespan they apply, it is not possible to determine the duration of events between successive states. This means that, with INT data as it is currently provided, it is not possible to represent continuous change, that is, change which occurs gradually over a time interval.

- Events in ITN data are therefore considered discrete, that is, they have a zero time duration.
• As there is no data regarding events provided with ITN data, then events are determined by analysing changes between successive states of a feature_state. Therefore, FEM events represent descriptions of change in the data between successive feature_states. There is no representation of a spatial_object_state as it undergoes change, other than its new representation as the new feature_state. The analysis of change between feature_states is described in section 4.4.

The name attribute gives the change a descriptive title, for example ‘Division’, or ‘Relocation’. The event type has no explicit valid time attributes, as these are specified in the spatial_object_states it references. These are the initial_state, which is the state as it was represented in the data before the event occurred, and the final_state, which is the state as it was represented in the data after the event occurred. Both the initial_state and final_state attributes are collections of references to spatial_object_states. For events such as creation, destruction, and change, these will reference single objects. However, for a splitting event, for example, the initial_state will reference the single spatial_object_state before the change, and the final_state will reference the objects that resulted from the split. The opposite is the case for a merger event: here the initial_state references the multiple spatial_object_states prior to the event, and the final_state references the single spatial_object_state created by the merger.

In the FEM model, the event type represents an event that alters the state of a spatial_object_state or its subtypes. The type attribute specifies whether the change is discrete or continuous. As stated earlier, for ITN data events are discrete with zero time duration. Continuous change could be represented if the valid_time values were more specific. For example, if a development event has occurred resulting in part of a road being redirected, then the valid_time value for the initial_state feature_state would have to correspond with the onset of the event, and the valid_time value for the final_state feature_state would have to correspond with the termination of
the event.

Events are not versioned, but form an integral part of a spatial_object_state.
Events are uniquely identified by their id and TOID attributes, and these correspond to the state_no and TOID attributes of the spatial_object_state referenced by its final_state attribute.

3.2.4 The transition type

A transition is formed by an aggregation of event objects and represents complex change in a feature's evolution. The events represented by the transition are those that have applied to a feature_state's constituent spatial_object_states, causing in it the changes that have created a new feature_state. The chronicle attribute of a transition is a collection of references to the event objects that represent the transition. Thus the chronicle attribute of the transition provides a change-based view of the feature's evolution. The initial_state attribute references the feature_state before any of the events represented by the transition have applied to it. The evolved_state attribute references the feature_state that has resulted from these events. The events referenced by the chronicle attribute may not apply to all of a feature_state’s constituent spatial_object_states, only those for which change has applied between the two feature_states referenced by the initial_state and evolved_state attributes of the transition.

It is not possible to determine the duration of the transition, as the available valid_time values for the feature_states referenced by the transition apply to the versions of the data representing the features and not reliable valid times for the features.

Unlike events, there is no description of the type of change that has occurred - this is defined by the events from which the transition is composed.
3.2.5 The evolved_feature type

At the highest level of organisation, the evolved_feature type maintains a history of a feature’s evolution. This type contains other state information, and is based on the structure of the generic object in the versioning system described by Chou and Kim (1988). It has two temporal attributes: initial_time and final_time, that span the earliest and latest valid times for all feature_states it references. An evolved_feature’s history of feature_states is organised as a temporally ordered list of references in the evolution_path collection attribute. The evolved_feature also contains state management information, as follows:

- state_count - the number of existing states for this feature
- next_state - the number to be assigned to the next state, incremented after each update
- default_state - used if no state or valid time is specified when querying

The evolved_feature can be considered a spatio-temporal hybrid object, encompassing a feature’s space-time extent in 2D + 1 dimensions.

3.3 Modelling real-world data

3.3.1 Alternative datasets

To model real-world data, the spatial_object_state type of the FEM model must be extended to include the geometric primitives and any other appropriate attributes. Although this research is constrained to use ITN data, there are other datasets that could be modelled by FEM, and two examples of these will now be described.
TIGER Line files

The US Census Bureau TIGER (Topologically Integrated Geographic Encoding and Referencing) system maintains a digital geographic database that provides complete coverage of the United States. Parts of the TIGER database are released periodically as TIGER database extracts, which provide geographic data for use in GIS applications. TIGER line files are such an extract [TIGER/Line Files Technical Documentation (2000)], and support programs in the US Department of Transportation and the Bureau of Transportation Statistics.

The TIGER database system is implemented in a RDBMS, and uses multiple point and line geometries to form what is referred to as chains which represent, for example, roads or streets. The configuration of these chains is maintained in a join table, but the TIGER system does not support features at a higher level of abstraction than complete chains, and so there are some features that are composed of multiple chains, with no explicit links to represent this.

OS MasterMap Topographic Data

The OS MasterMap Topography Layer [Ordnance Survey MasterMap Topography Layer (2008)] is implemented in a relational structure and uses point, line, and polygon geometries to represent real-world objects such as buildings, streets, and letter boxes. Complex features are represented as multiple geometries, but there is no explicit link between complex features and their constituent geometries, and updates are destructive.

Both these datasets have a comparable structure to the ITN data, and could be represented within the FEM model.
3.3.2 ITN data

ITN data provides a topologically structured representation of the UK’s driveable roads. The representation is continually updated, and as such can be regarded as composed of evolving road features. Road features in ITN data are composed of links and nodes (see figure 3.2), and changes to these components represent a change in the road feature. The basic unit of the road feature is the roadlink. Roadlinks are comprised of a polyline geometric primitive with a roadnode object at each end. A roadnode object has a point geometric primitive. A road feature is an aggregate of roadlink objects. The road feature does not contain the road’s geometry, only a reference to its constituent roadlink objects. Roadlinks, roadnodes, and roads all have the attribute change history in the standard ITN data, which is a collection data type. Each element of this collection has a date and description, which is either new or modified. No other information on the changes is recorded.
Although the ITN data contains spatial objects, it remains essentially relational in nature, meaning that aggregation of roadlinks is represented by relational joins, and roadnode objects relating to roadlink objects are similarly referenced in another join table. The version information provided means that the frequency of change can be seen, but previous versions of features cannot be retrieved from the data.

3.3.3 Events in ITN data

The events represented by the FEM model in ITN data were based on observed changes between different versions of the data. These observations were made by visualising features using the preview feature of an Oracle utility called MapBuilder[Murray (2006)]. Events were represented for links of a road feature only, that is, events are not described for nodes. The analysis of events (except creation and destruction) was determined by changes to the positions of a roadlink_typ’s roadnode_typ objects. The following events are represented by the FEM model for ITN data. Detection of these events is described in section 4.4.

- Creation - an event that creates a new link. A creation event references the created link via its final_state attribute. It has a null initial_state attribute.

- Destruction - an event that results in the destruction or removal of a link. A destruction event references the removed/destroyed link via its initial_state attribute. It has a null final_state attribute

- Division - an event that divides a link into two or more links. This is caused by the creation of a new node in between the two existing nodes of a link. Figure 3.3 shows such an event. Here, state 1 of a link undergoes a division event by the creation of node C between nodes A and B. One of the two links thus created retains the TOID of
the original link. In the FEM model, therefore, there are two objects involved in the event: a new link and an updated link (shown in the diagram as Link State 2). (This representation of a division event is also used by Vidal and Rodriguez (2005).) The new link will have a creation event associated with it, but the division event will reference the original link via its initial_state attribute, and both the updated and new links via its final_state attribute.

- Fusion - an event that merges two or more links into one link. This is caused by a node that is the end of one link and the start of another being removed. In figure 3.4 node B of state 1 has been removed in state 2, creating the new link. One of the TOIDs from the two links that have been merged is retained in the newly formed link, and one is lost, that is the other link is destroyed. The fusion event will reference the two links in state 1 via its initial_state attribute, and the single link forming state 2 via its final_state attribute. The destruction event will reference the discarded link via its initial_state attribute.

- Relocation - an event that results in a link moving to a new location. This occurs when both nodes of a link change their position, as shown in figure 3.5.
Figure 3.4: Fusion event in ITN data

Figure 3.5: Relocation event in ITN data
• Redirection - an event that results in a link being redirected. This occurs when one node only of a link changes its position, as shown in figure 3.6.

3.3.4 Modelling ITN data with FEM

Figure 3.7 shows a class diagram of the specialised classes of the FEM model necessary to model ITN data. The polyline (link) and point (node) geometric primitives are represented in the `roadlink_typ` and `roadnode_typ` types, respectively. In addition to the polyline spatial attribute, the `roadlink_typ` includes other attributes specific to a link. These are `length`, `natureofroad`, `is_part_of`, and `nodes`. The `is_part_of` attribute is an association that links the `roadlink_typ` to its parent `feature_state` object. The `nodes` attribute is a collection of references to `nodetopo` objects. The `nodetopo` is an extra type necessary to maintain topological relationships, and contains a reference to a `roadnode_typ` and an `orientation` attribute, which denotes at which end of the link the `roadnode_typ` is positioned. The `roadnode_typ` subtype simply adds its point geometric primitive.
Figure 3.7: Class diagram showing specialised classes for modelling ITN data
To demonstrate how FEM models typical change in the ITN data, four typical transitions between feature states will be modelled and described, with the object instances and interactions shown using object diagrams.

1. A feature_state that has been extended by the creation of two new links.

2. A feature_state that has one of its links split by a new node.

3. A feature_state that has two of its links fused into one link.

4. A feature_state that has one of its links redirected, and one of its links relocated.

**Transition 1 - feature_state extended by the creation of two new links**

This transition is modelled by the object diagram shown in figure 3.8. The starting point for the transition is the evolved_feature object called EvFeat (top left), showing initial_time and current_time values of 06-SEP-04, meaning that only one feature_state is represented in its evolution_path attribute, shown as the association value of REF STATE_1. This feature_state, called simply STATE_1, shows the valid_time value of 06-SEP-04. STATE_1’s configuration is shown as a single link, represented by the value REF LINK_1 STATE_1 for its elements attribute. This link is shown, called LINK_STATE_1, and is an instance of the roadlink_typ type, also with the valid_time value of 06-SEP-04. Two event objects show the creation of new links: CREATION_1 represents the creation of the roadlink_typ object called LINK_2 STATE_1 which has the valid_time value 09-JUNE-06, and the final_state value for the association between CREATION_1 and LINK_2 STATE_1 is shown as the value REF LINK_2 STATE_1; CREATION_2 represents the creation of the roadlink_typ object called LINK_3 STATE_1 which also has
the *valid_time* value 09-JUNE-06, and the *final_state* value for the association between `CREATION_2` and `LINK_3` *STATE 1* is shown as the value `REF LINK_3 STATE 1`. These two *event* objects are represented in the *transition* object called `EVOLVER`. Three association values are shown for this object: *initial_state* has the value `REF STATE_1`, *evolved_state* has the value `REF STATE_2` (referencing the *feature_state* being created by this transition, called simply `STATE_2`), and the *chronicle* attribute shows the values `REF CREATION_1` and `REF CREATION_2`, representing the aggregation of the *events* in this *transition*. The newly created link objects are reflected in the configuration of the new *feature_state*, `STATE_2`, by the values `REF LINK_1 STATE 1`, `REF LINK_2 STATE 1`, and `REF LINK_3 STATE 1` of its *elements* attribute. The *evolved_feature* has been updated, called `EvFeat_updated`, and has the new *current_time* value of 09-JUN-06 and the *evolution_path* values of `REF STATE_1`, `REF STATE_2`.

**Transition 2 - *feature_state* that has one of its links split by a new node**

This transition is modelled by the object diagram shown in figure 3.9. The starting point for the transition is, again, the *evolved_feature* object called `EvFeat` (top left), showing *initial_time* and *current_time* values of 06-JAN-04 and 06-SEP-04 respectively. For simplicity, only the most recent of its *feature_states* is shown in its *evolution_path* attribute, shown as the association value as `REF STATE_1`. This *feature_state*, called `STATE_1`, shows the *valid_time* value of 06-SEP-04, corresponding to the *current_time* value of `EvFeat`. `STATE_1`'s configuration is again shown as a single link, with the value `REF LINK_1 STATE 1` for its *elements* attribute. The roadlink_typ object, `LINK_1 STATE_1`, also has the *valid_time* value of 06-SEP-04. The division *event*, called `DIVISION`, splits `LINK_1 STATE 1` by the addition of a node along its polyline geometry. The new node is not shown in the diagram, as in FEM events are not attributed to nodes. `DIVISION`
Figure 3.8: Object diagram modelling a road feature extended by the creation of two links in ITN data.
shows its \textit{initial\_state} value as REF LINK\_1 \texttt{STATE\_1}, and its \textit{final\_state} values as REF LINK\_1 \texttt{STATE\_2} - the updated object, and REF LINK\_2 \texttt{STATE\_1} - the newly created link. Both these objects have the \textit{valid\_time} value of 09-JUN-06. The \textit{event} object called CREATION represents the creation of LINK\_2 \texttt{STATE\_1} and shows the \textit{final\_state} attribute value of REF LINK\_2 \texttt{STATE\_1}. The new \textit{feature\_state} called \texttt{STATE\_2} is shown with the new configuration of its \textit{elements} attribute values. As in the previous example, the two \textit{event} objects are represented in the \textit{chronicle} attribute values of the \textit{transition} object called EVOLVER, along with this objects \textit{initial\_state} and \textit{evolved\_state} values of REF \texttt{STATE\_1} and REF \texttt{STATE\_2} respectively. The updated \textit{evolved\_feature} object, EvFeat\_updated, shows the updated \textit{current\_time} value of 09-JUN-06, and \textit{evolution\_path} values of REF \texttt{STATE\_1}, REF \texttt{STATE\_2}.

**Transition 3 - feature\_state** that has two of its links merged into one link

This transition is modelled by the object diagram shown in figure 3.10. Again, EvFeat is the starting point for the transition showing the \textit{initial\_time} and \textit{current\_time} values of 06-SEP-04, meaning that only one \textit{feature\_state} is represented in its \textit{evolution\_path} attribute, shown as the association value of REF \texttt{STATE\_1}. The \textit{feature\_state} \texttt{STATE\_1} shows the \textit{valid\_time} value of 06-SEP-04, and its configuration is shown by the values REF LINK\_1 \texttt{STATE\_1} and REF LINK\_2 \texttt{STATE\_1} of its \textit{elements} attribute. The thus referenced roadlink\_typ objects, LINK\_1 \texttt{STATE\_1} and LINK\_2 \texttt{STATE\_1}, have the \textit{valid\_time} value of 06-SEP-04. The event called FUSION merges LINK\_1 \texttt{STATE\_1} and LINK\_2 \texttt{STATE\_1} by the removal of a node that is the end node of LINK\_1 \texttt{STATE\_1} and the start node of LINK\_2 \texttt{STATE\_1}. FUSION shows its \textit{initial\_state} values as REF LINK\_1 \texttt{STATE\_1} and LINK\_2 \texttt{STATE\_1} and its \textit{final\_state} value as REF LINK\_1 \texttt{STATE\_2}, the updated link object, which has the \textit{valid\_time} value of 09-JUN-06. The \textit{event
Figure 3.9: Object diagram modelling a division event
object called DESTRUCTION represents the removal of LINK_2 STATE 1 and shows the initial_state attribute value of REF LINK_2 STATE 1. The new feature_state called STATE_2 is shown with the new configuration shown by its elements value. The EVOLVER transition and EvFeat evolved_feature objects are updated accordingly.

**Transition 4 - feature_state that has one of its links redirected, and one of its links relocated**

This transition is modelled by the object diagram shown in figure 3.11. The evolved_feature and initial feature_state are present and have the same attribute values as in the previous scenario. The event REDIRECTION represents the redirection of the roadlink_typ object called LINK_1 STATE 1, forming the roadlink_typ object LINK_1 STATE 2 which has the valid_time value 09-JUN-06. The initial_state and the final_state values for REDIRECTION are shown by the values REF LINK_1 STATE 1 and REF LINK_1 STATE 2 respectively. Similarly, the event called RELOCATION modifies LINK_2 STATE 1 forming LINK_2 STATE 2, which also has the valid_time value 09-JUN-06. RELOCATION shows the appropriate initial_state and final_state values. Again, the new feature_state, transition, and evolved_feature objects are updated accordingly.

**Summary**

This chapter has established the foundation of the FEM spatio-temporal data model, and shown how it applies to some real-world data. The design principles highlighted in the first section mean that the model is able to represent an absolute-hybrid spatio-temporal model with just five core types. Further, due to the expressiveness of object-orientation, these types are able to represent three levels of state versioning and both primitive and compound events. The type descriptions given in the second section show
Figure 3.10: Object diagram modelling a fusion event
Figure 3.11: Object diagram modelling redirection and relocation events
that the data structures are simple yet powerful enough for this representation. Two kinds of versioning have been adapted from versioning mechanisms used in CAD/CAM database systems: type versioning, where the spatial object type is extended to include the geometric primitives appropriate to the application domain; and configuration management, where spatio-temporal configurations are maintained and organised by the evolved _feature type. The final section demonstrates that the model can effectively represent complex, asynchronous change in real-world data, in this case ITN data.

Relating the model to the criteria for an optimal data model set out in table 3.1, it can be seen at this stage that the model provides structural validity in its representation of ITN data, simplicity in requiring only five core types, and expressibility in the way objects and relationships are defined. Further, the model has a degree of shareability in that it can be implemented in any OO or OR environment, and can be clearly expressed diagrammatically. The extension of the spatial_object_state type to support ITN data and the relevance of other datasets that could be represented within the model demonstrate its extensibility. The other criteria for an optimal model, that of non-redundancy and integrity, will be assessed in the evaluation chapter.

The next chapter will describe the implementation of an FEM STDBMS in Oracle Spatial, and the subsequent data entry, update, and event analysis.
Chapter 4

Implementation of the FEM STDBMS

This chapter will describe the implementation of a STDBMS based on the FEM model. The first section describes the benefits of an object-relational implementation. Section 4.2 describes the creation of the base state, how Oracle SQL is equipped to implement the types of the schema, and the creation of FEM objects using custom Java classes. Section 4.3 describes the addition of updated features into the FEM STDBMS, and the mechanisms that maintain histories and feature configuration. The final section provides a discussion on events, and explains how ITN events are determined and represented.

4.1 The benefits of the object-relational model

The DBMS to be used for this research, Oracle Spatial, is object-relational (OR), and as such supports objects known as abstract data types (ADTs). Spatial ADTs can have spatial and aspatial data associated with them, and allow geographic features to be abstracted independently [Rigaux et al. (2002), Voisard and David (2002)]. ADTs also offer the ability to extend spa-
tial objects in the form of user defined types (UDTs) to include versioning information and temporal attributes (to form states), which are important in the representation of evolving features. Further, the OR model includes collections, meaning that attribute histories can be recorded by combining a time element with a data value or object. The OR model also means that aggregation and other associations can be modelled explicitly using pointers, called REFs, instead of the more costly (in performance terms) and less expressive relational joins. These benefits mean that the framework exists within Oracle Spatial to design and implement a model to effectively implement an absolute-hybrid spatio-temporal representation.

4.2 Creating the base state

The first stage of the implementation was to create an initial, or base state of FEM objects from the standard ITN data. This data was supplied by Ordnance Survey as a complete set of data in its relational structure, and was first entered into a Oracle database in its standard form. This data was organised principally in the following tables:

- **ROAD** - contains the feature attributes but no spatial attribute (except the feature’s minimum bounding rectangle (MBR)), including a `versiondate` attribute, denoting the feature’s valid time.

- **ROADLINK** - contains link geometric primitive in the form of a polyline, and associated attributes, including a `versiondate` attribute, denoting the link’s valid time.

- **ROAD_NETWORKMEMBER** - this is a join table, and represents a road feature’s configuration by associating a link `TOID` attribute with a road `primary_key` attribute.

- **ROADNODE** - contains node geometric primitive as a point object,
<table>
<thead>
<tr>
<th>FEM Type</th>
<th>Table name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolved_feature</td>
<td>EVOLVED_FEATURE_TAB</td>
</tr>
<tr>
<td>Feature_state</td>
<td>FEATURE_STATE_TAB</td>
</tr>
<tr>
<td>Roadlink_typ</td>
<td>LINKSTATE_TAB</td>
</tr>
<tr>
<td>Roadnode_typ</td>
<td>NODESTATE_TAB</td>
</tr>
<tr>
<td>Transition</td>
<td>TRANSITION_TAB</td>
</tr>
<tr>
<td>Event</td>
<td>EVENT_TAB</td>
</tr>
</tbody>
</table>

Table 4.1: FEM object tables

and associated attributes, including a *versiondate* attribute, denoting the node’s valid time.

- **ROADLNK_DIRECTEDNODE** - a join table, associating a node object’s *TOID* with a link object’s *primary key*, and also specifies which end of the link the node is positioned.

### 4.2.1 Implementing the FEM schema

The FEM schema was implemented in Oracle Spatial using Oracle SQL. Oracle SQL provides extensions to SQL to exploit Oracle’s object-relational features. These extensions are in the form of new data definition language (DDL) commands and data manipulation language (DML) commands. Oracle SQL DDL commands allow the creation of object types (or ADTs), nested tables, and arrays, defining table columns of object types, and creating object tables. Oracle objects themselves can be part of other objects, either by using REFS or in collection attributes. Oracle SQL DML allows the querying and updating of objects and collections, and the manipulation of REFs.

The FEM schema was set up using DDL commands according to the types specified in figure 3.1. An object table was set up for each object type, as shown in table 4.1. (Note that although the OO feature of substitutability allows both *roadlink_typ* and *roadnode_typ* objects to be stored in one object table of type *spatial_object_state*, this was not done due to spatial indexing.
requirements. (A bug in Oracle Spatial means that a spatial column in a subtype cannot be specified in the metadata table to enable spatial indexing (see section 4.3)).

4.2.2 Oracle objects and JDBC strongly typed interfaces

The process of creating and entering FEM objects was accomplished using Java. Java is a powerful object-oriented programming language, and provides a secure, portable application development environment. Oracle provides access to Oracle object data via JDBC (Java database connectivity), and Oracle JDBC strongly typed interfaces can provide tight integration between Oracle and Java objects by mapping Oracle database objects to custom Java classes.

Having the database object as a Java object means that proper setter and getter methods can be used to manipulate attributes with minimal use of SQL, and overcomes many of the inefficiencies when mapping a declarative language such as SQL, to an imperative language such as Java (the impedance mismatch) [Connolly and Begg (2002)]. Oracle provides a convenient utility, called JPublisher [Menon (2005)], that can translate Oracle types into corresponding Java classes. Classes generated by JPublisher can be extended to add additional functionality if required, and the resultant custom classes can be used from the calling Java application to perform DMLs [Menon (2005)].

Oracle objects are read from and written to the database using the getORAData() and setORADATA() methods of the Oracle JDBC ORAData interface. Examples of these are given below:

The following code snippet shows how a roadlink_typ object is read from the database and cast to its corresponding RoadlinkTyp java object:

```
String selectStmt = "select value(a) from LINKSTATE_TAB a where a.TOID = ?";
PreparedStatement rpstmt = conn.prepareStatement(selectStmt);
```
The following code snippet shows how the Java RoadlinkTyp object is written to the database:

```java
PreparedStatement pdstmt = conn.prepareStatement("insert into LINKSTATE_TAB values (?, ?)";
((OraclePreparedStatement)pdstmt).setORAData(1, rltyp);
pdstmt.executeUpdate();
```

Other aspects of the generation of the custom classes are important. Collections are generated as objects in their own right, as are REFs. This means, that for example the elements attribute of a roadlink_typ object in Oracle will be translated as an object which is a collection of Java objects each representing a REF. These Java ref objects provide a getValue() method to access the referenced objects and its attributes.

The JPublisher utility was used to create the corresponding Java custom classes for the implemented FEM types.

### 4.2.3 Creating FEM objects

Although the use of strongly typed interfaces is designed to minimise the use of SQL, this cannot be avoided initially as the Oracle objects do not yet exist. To create them, the ITN data must be transformed from its relational structure to the FEM structure, and was achieved in the following stages:

For a each feature TOID:
• A feature_state object was created, with a null configuration (elements attribute) and then an evolved_feature object was created, with a reference to this feature_state in its evolution_path attribute.

• The feature’s configuration was determined using SQL select statements (embedded in JDBC Statement and PreparedStatement objects) by querying the ROAD_NETWORKMEMBER table.

• The appropriate roadlink_typ objects were instantiated for the feature’s configuration, and its elements attribute updated with corresponding REFs.

• The link’s node configuration was determined by querying the ROADLINK_DIRECTEDNODE table.

• The appropriate roadnode_typ and nodetopo objects were instantiated for the roadlink_typ, and its nodes attribute updated with corresponding REFs.

The creation of a FEM object was achieved as follows:

1. The ITN data was read from the relational tables using SQL select statements (embedded in JDBC Statement and PreparedStatement objects).

2. The attributes retrieved from the select statements (in the ResultSet object) were assigned to Java variables.

3. These Java variables were then used to instantiate a Java object corresponding to the required Oracle object.

4. The Java object was into the FEM database (as described in section 4.2.2).
Using these methods, the base state of the FEM STDBMS was established. All objects at this stage are state 1, and no events have yet been described. The next two sections will describe the insertion of a set of ITN change only updates (COU) data into the database in a non-destructive manner, the subsequent analysis of the changes that occurred in the data between updates, and their representation in the database.

4.3 Updating the base state

Typically, updated ITN data is supplied in the previously mentioned COU format, and would be inserted into the standard tables by updating the attributes and deleting the outdated features. The following section will describe how this data can be inserted into the FEM STDBMS.

The basis of inserting the COU data is essentially the same as the insertion of the base state, but with more analysis necessary to maintain feature configuration. The system will store all states of each object (feature_state, roadlink_typ, and roadnode_typ objects) but not duplicate any. This is important for storage requirements, and reflects the fact that a road feature’s links will not necessarily undergo change at the same time. The insertion of this data followed the following stages:

- The standard COU data was first entered in the same way as a base state, but into separate tables. These were named similarly to the base state tables, but each name was prefixed with ‘UPDATE_’. The description of these tables is as described in section 4.2, and their names were as follows:

  - UPDATE_ROAD
  - UPDATE_ROADLINK
  - UPDATE_ROAD_NETWORKMEMBER
  - UPDATE_ROADNODE
For each COU feature **TOID:**

- The FEM database was checked to determine if this feature was present. If it was not, a new feature was created as state 1, as if inserting a base state feature, and the update finished. If the feature was present, the appropriate state number was determined by querying the corresponding evolved_feature object, and the other numerical attributes of this object were incremented (**state_count**, **next_state**, and **default_state**). The update would then continue, as follows.

- A feature_state object was created for the determined state number, and the evolution_path attribute of its evolved_feature updated with the corresponding REF.

- The update feature’s configuration was checked, as described in the pseudocode below, and the corresponding roadlink_typ objects created and inserted, and the feature_state’s elements attribute updated with the corresponding REFs.

- Similarly, the update link’s node configuration was checked, the appropriate roadnode_typ and nodetopo objects created, and the roadlink_typs nodes attribute updated.

- Delete COU tables

When creating a new feature_state object as an update for an existing feature_state, several factors affect the correct determination of its configuration, and whether or not to create new roadlink_typ object. This is because not all the feature’s constituent links may have been updated, and so it will not be necessary to create roadlink_typ objects for links that have not changed. The following pseudocode details this process and refers to the following variables:
**COU** - the COU data stored in the tables described earlier

**FEM** - the FEM database system

**VERSIONDATE** - the versiondate attribute of a link in the COU data

**VALID_TIME** - valid_time attribute of a roadlink_typ object in the FEM database

---

**Retrieve the TOIDs for the Feature's Constituent Links by Querying the Update_Road_NetworkMember Table**

For each link TOID:

```{  
IF LINK TOID PRESENT IN COU BUT NOT IN FEM //that is, we do not already have a version of this link
  INstantiate roadlink_typ (STATE 1) and a REF}

IF TOID PRESENT IN COU AND FEM //that is, we do have a version of this link in the FEM database

{  
  IF VERSIONDATE = VALID_TIME //it is the same version
    CREATE REF ONLY
  ELSE IF VERSIONDATE > VALID_TIME //it's a later version than the one we have
    Instantiate roadlink_typ and ref}

IF LINK NOT PRESENT IN COU BUT PRESENT IN FEM //the link data is not present in the COU data (UPDATE_ROADLINK table), but its TOID is referred to in the
In this way, only new `roadlink_typ` objects are created and stored in the FEM database, but REFs are created for all the feature’s links and inserted into its `elements` collection attribute. A similar algorithm determines the correct node configuration for a `roadlink_typ` object, and updates its `nodes` attribute.

The next section will detail how events are determined and `event` objects created.

## 4.4 Determining events

As discussed in the introductory chapter, an absolute-hybrid spatio-temporal implementation must include the representation of events. In this regard, an issue with the implementation of the FEM system using ITN data is that this data has been captured with no regard for events, and so changes between states are not explicitly captured. Section 3.3.3 describes the events that can be captured by analysing the differences between versions of the features in the FEM database system. To enable this representation an analysis function was executed to identify which objects of which features underwent which types of change. This facilitated the creation of the appropriate `event` and `transition` objects to complete the representation. Oracle objects to be analysed were read from the database in Oracle SQL and instantiated as Java objects prior to analysis, as described in section 4.2.2.
4.4.1 Event determination algorithm

The types of events that can occur to links of road features in ITN data were
described in section 3.3.3. As mentioned in that section, events apply to links
only, and not to node objects. Further, events are analysed by determining
changes to the positions of links’ nodes, and no analysis is made of changes
to the coordinates of vertices within links. There is also no explicit reference
given to node objects in the event representation. These are limitations to
the representation of events in the implementation, and mean that changes
to link’s node configuration can only be accessed by manually querying its
nodes attribute and comparing the node configurations of successive link
states. Internal changes to the vertices of links can only be assessed with a
visual overlay of successive states.

In order to determine and represent ITN events in the FEM database,
an event determination algorithm was run after each update. This algorithm
compared the new state of a feature with its previous state, and created event
objects according to certain criteria, which are detailed in the pseudocode for
this function, as follows:

```plaintext
variable: NEW_STATE_LINKS - an array of the roadlink_typ TOIDs
for the new feature state
variable: UP_STATE_LINKS - an array of the roadlink_typ TOIDs for
the updated feature state

for each TOID in NEW_STATE_LINKS {
    if TOID not present in UP_STATE_LINKS //the link is com-
        pletely new to the FEM database system
        CREATION EVENT for the link identified by this
        TOID in NEW_STATE_LINKS
    else if TOID present in UP_STATE_LINKS and its STATE_NO
        is higher
        //other event criteria
}
```
run ANALYSIS FUNCTION //determine Relocation, Redirection, Division, Fusion events 

for each TOID in UP_STATE_LINKS{
    if TOID not present in NEW_STATE_LINKS //the link no longer exists
        DESTRUCTION EVENT for the link identified by this TOID in UP_STATE_LINKS
}

The analysis function referred to in the above pseudocode was used to compare a new link with its previous state. The analysis function determined redirection, relocation, division, and fusion events. At the core of this function is a method of type Boolean, that takes two roadnode_typ objects as arguments and analyses their point attribute geometries by comparing the coordinates. The method returns true if the point attribute coordinates are different, that is the node object has changed its position.

The analysis function is run when the link for a feature_state is present in its preceding state, and has a higher state number. The function compares each node of a link with the nodes of its previous state, and thereby makes symantic distinctions between events according to the following criteria:

• if both of the node TOIDs for the links being compared match
    – If both nodes have changed their position: Relocation
    – If only one of the nodes has changed its position: Redirection

• if one of the node TOIDs for the links being compared is different
    – If the length of the link is shorter: Division
    – If the length of the link is longer: Fusion

Event objects are created according to these semantic distinctions.
4.4.2 Spatial indexing

The final part of the implementation was to create the spatial indexes. This is necessary because even though the associations between objects provides navigational access to attributes when querying, spatial operations require spatial indexing. There are no special requirements for spatial indexing in Oracle Spatial with the FEM implementation. All that is necessary is to update the metadata table (USER_SDO_GEOM_METADATA) with the columns of the object tables that contain the geometries to be indexed, and create the indexes.

Summary

This chapter has described the implementation of a STDBMS based on the FEM spatio-temporal data model, using Oracle Spatial and it’s OR capabilities. This has enabled an absolute-hybrid spatio-temporal representation by abstracting the spatial objects in the ITN data in a more natural and expressive way, and eliminating feature succession, whilst retaining a feature’s spatio-temporal configuration. The structure that the FEM model imposes on the data means that both configuration-based and chronicle-based views are accessible from a single container.

The next chapter will describe the accessing and querying of FEM objects, give some examples of spatio-temporal queries, and describe the various methods for executing them.
Chapter 5

Querying the FEM STDBMS

This chapter will describe the advantages of an OR implementation of the FEM data model in terms of data access and querying. In section 5.1 the nature of accessing FEM objects and their attributes will be described, and in section 5.2 the specific advantages of the FEM data structure will be detailed. Section 5.3 will outline some example queries that can be performed on the FEM STDBMS and the methods for executing these queries.

5.1 Accessing spatio-temporal data

The three basic components of spatio-temporal data are spatial, attribute, and temporal data. From a querying perspective, these data reside in a multi-dimensional global data space, whose axes are determined by the query’s predicates, which are in turn determined by their domain. Temporal and spatio-temporal queries attempt to restrict this space to a specific search area according to temporal and spatio-temporal predicates along the spatial, attribute, and temporal axes [Langran (1992)].

The following are some basic kinds of spatio-temporal queries:

1. Simple temporal query, for example what is the state of a feature at time $t$?
2. Temporal range query, for example what happens to a feature over a given period?

3. Simple spatio-temporal query, for example what is the state of a region at time $t$?

4. Spatio-temporal range query, for example what happens to a region over a given period?

Queries 2, 3, and 4 are range queries, their search space being a range in the multi-dimensional data space. Queries 1 and 3 fix time: query 1 controls attributes (to the domain of that feature), and query 3 specifies location (to the region) [Langran (1992)].

Figure 5.1 shows the search space for these four spatio-temporal queries. Here, an FEM feature_state occupies a zero-dimensional search space because its attributes have single values along the axes of the data space, as shown in figure 5.1 (a). A feature_state's temporal trajectory is shown by a vector, as shown in figure 5.1(b), and at each location on the time line the
state's attributes have a single value. These two queries can be considered temporal queries, that define the data space in 2 dimensions by attributes and time. Spatio-temporal queries define a geometric data space, that can describe points, lines, and polygons, and their relationships. This space models an object's spatial extent, changes in its entities, and its space-time evolution. In the case of an FEM feature state, this space will be two-dimensional in the case of a simple spatio-temporal query, as in figure 5.1 (c), and three-dimensional in the case of a spatio-temporal range query, as in figure 5.1 (d)[Langran (1992)].

5.2 Accessing FEM objects

The FEM data model represents features, spatial entities, and events as interrelated objects, forming a graph. The relationships between objects and events is established using references (REFs). When accessing these objects, an application can firstly access an initial object or group of objects that are of particular interest, and then traverse the graph using the references to access related objects and their attributes, performing operations on them if required. This kind of access is known as navigational access, as opposed to the associative access necessary in a relational model, where relationships are created using foreign keys, and where querying multiple tables requires complex joins. Accessing objects in an object-based system also means that you can retrieve an object from the database along with all other objects connected to it in one round-trip [Agrawal et al. (2007)].

In the FEM data model, time is treated as an attribute of the spatial objects, and this allows us to access a feature’s evolution by querying a linearly ordered list of references to a feature’s states (the evolution_path attribute of the evolved_feature object), each of which comprises its attributes and validity dates. By accessing the transition object referenced by a feature_state’s evolver attribute, a description of the changes to the feature from its previous
state can be obtained. The references used in the FEM structure overcome a problem that could be encountered with traditional GIS models where spatial and attribute data are stored separately, meaning that queries have no direct access to both forms of data [Langran (1992)]. FEM’s OO model also allows SQL objects to be mapped to corresponding Java objects, as discussed in the previous chapter, meaning that objects can be loaded by Java applications and the graph similarly traversed.

5.3 SQL 2003 and Oracle SQL

The universal language for querying and manipulating data in relational database systems is SQL (structured query language). SQL allows users to interact with data at the logical level and provides an interface to the database system [Lorentz and Gregoire (2003)]. SQL came into use at the same time the relational model was introduced in the early seventies. Since then the language has been developed and extended, and attempts have been made to include the temporal dimension. Researchers have attempted to produce a standard temporal query language, which produced TSQL2 [Snodgrass (1995b)] although to date this has not been widely adopted. Rather, SQL has developed to include statements that can manage the demands of modern database systems, including objects.

The latest SQL standard was formalised in 2003 and is referred to as SQL 2003 [ISO (2003)] and has been incorporated into Oracle SQL [Lee (2003)]. This means that in addition to processing relational data, Oracle SQL can construct the complex data structures used in the FEM representation, including objects with full support for inheritance, collections and nested tables, and REFs. Querying these data structures requires the use of some specific operators and functions, which will now be described.
5.3.1 Querying REFs

Accessing an object that is pointed to by a REF attribute is achieved in two ways. If the whole object is being retrieved then the DEREF function is needed. If DEREF is not used, then an object identifier (OID) will be returned by the query. So, this query:

\[
SELECT P.MODIFIER FROM LINKSTATE_TAB P \\
WHERE P.TOID = 'osgb400000013049080';
\]

will return an OID, whereas this query:

\[
SELECT DEREF(MODIFIER) FROM LINKSTATE_TAB P \\
WHERE P.TOID = 'osgb400000013049080';
\]

will return the complete event object instance showing all attributes.

If only attributes of the referenced object are required, and not the entire object, then these can be accessed using the dot operator. For example, the following query retrieves the name attribute of the event object referenced by a roadlink_typ:

\[
SELECT P.MODIFIER.NAME FROM LINKSTATE_TAB P \\
WHERE P.TOID = 'osgb400000013049080';
\]

5.3.2 Querying subtypes

Subtypes can be accessed in the same way as other objects, unless they are stored in an object table which has been declared a table of the supertype of the object. When a subtype is stored in such a way, the extra attributes of
the subtype are stored in hidden columns of the table and will not be visible to the system under normal querying. These attributes are made visible using the TREAT function. This means that if roadlink_typ objects were stored in an object table of type spatial_object_state, then inherited attributes can be accessed using simple select statements, but the extra attributes of the subtypes need to be made visible using TREAT. For example, the following query would retrieve the polyline attribute from a roadlink_typ object stored in such a way:

\[
\text{SELECT TREAT(VALUE (P) AS ROADLINK_TYP).POLYLINE FROM SPATIAL_OBJECT_STATE_TAB P WHERE P.TOID = 'osgb400000013049080';}
\]

Note that although this query has been used as an example, this kind of storage was not used in the implementation due to issues with indexing (see section 4.2.1).

### 5.3.3 Querying nested tables

Accessing nested tables requires the use of the TABLE command to identify the table, and, as was the case with the FEM implementation, the COLUMN_VALUE command to identify the column if the target table is an object table. In the following example, all objects in the elements nested table of a feature_state are retrieved:

\[
\text{SELECT L.COLUMN_VALUE FROM FEATURE_STATE_TAB P, TABLE(P.ELEMENTS) L WHERE P.TOID = 'osgb400000013332209';}
\]

It can be seen here that the TABLE command identifies the nested table as the elements attribute of the feature_state, and assigns it the alias L, which is used by the COLUMN_VALUE command to identify the column.
Operator | Definition
---|---
= | equal to
< | before
> | after
BETWEEN | between

Table 5.1: SQL temporal operators

heading. However, in this instance the nested table is a table of references to \textit{spatial\_object\_state} objects and so this query will return a list of object references. Declaring the nested table to be of a supertype means that it can contain objects of the supertype and any of its subtypes, but also means that the TREAT function must be used to access subtype objects and their attributes, as is the case in some of the example queries detailed in section 5.3.3.

5.3.4 Temporal and spatio-temporal operators

A GIS must include spatial operations in its range of functions, such as proximity and distance functions, relationship functions, and buffering. The database system used for the implementation of the FEM STDBMS, Oracle Spatial, supports such operations, and these operations and functions can be executed in SQL [Kothuri (2005)]. Further, a temporal GIS must provide temporal operators that can combine with spatial predicates to form spatio-temporal operators. Table 5.1 shows some temporal operators which are useful when formulating temporal and spatio-temporal queries.

Combining temporal operators with certain spatial operators enabled the formulation of spatio-temporal queries. The spatial operator that was used to restrict queries to a specific area was SDO\_INSIDE. This function identifies all geometries that fall within the boundary of a specified query geometry. The structure of the function is as follows:
$SDO_{INSIDE}$

$($
TABLE GEOMETRY ($SDO$ GEOMETRY)
QUERY GEOMETRY ($SDO$ GEOMETRY)
$)$

$= ‘TRUE’$

Where

- TABLE GEOMETRY is the column name of the table containing the geometries being queried
- QUERY GEOMETRY is the query area.

### 5.3.5 Change expressions

The inclusion of events in the FEM data model allowed the construction of change expressions by combining event predicates with temporal operators. A change expression can be included in the *where clause* of a query to restrict the results to specific events and specific times or time periods. An example of the use of a change expression would be ‘Retrieve all features that underwent a division event between two times’. The change expression for this query would be as follows:

WHERE MODIFIER.NAME = ‘Division’ AND VALID_TIME BETWEEN ‘12-JUL-06’ AND ‘23-SEP-07’;

Here MODIFIER.NAME refers to the event name from the modifier attribute of a roadlink_typ object, and VALID_TIME refers to this attribute of the roadlink_typ. The BETWEEN temporal operator specifies the time range.
5.4 Example queries

It can be seen from the descriptions of how data can be accessed in the FEM model that the four fundamental types of spatio-temporal query can be viewed from either a configuration-based or an event-based perspective. Examples of each will now be provided, along with the various ways the data can be accessed. However, certain aspects of querying are common to both views. These are as follows:

- both configuration-based and event-based data are accessible from the evolved_feature object, via the evolution_path attribute by traversing the graph via their respective routes
- as feature_states and transitions share the same identifier, these objects can be queried directly if the TOID is known (TOID in a feature_state corresponds to the transition_id for that feature)
- intervals are implicit in the evolution_path as feature_states have a valid_time attribute and the list is temporally ordered, but are subject to the state of partial ignorance inherent in these values.

In the following sections examples of FEM queries are given based on the fundamental spatio-temporal queries outlined in section 5.1.

5.4.1 Configuration-based queries

1. What is the configuration of a feature at time t?

   SELECT TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP). POLYLINE FROM FEATURE_STATE_TAB p, TABLE(p.ELEMENTS) L WHERE p.TOID = 'osgb4000000013332209' AND P.VALID_TIME = '23-JUL-07';

2. How did a feature’s configuration change over a given period?
3. What configuration did the features composing a region have at time t?

```sql
SELECT TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP).POLYLINE FROM FEATURE_STATE_TAB p, TABLE(pELEMENTS) L WHERE p.TOID = 'osgb400000013332209' AND p.VALID_TIME BETWEEN '12-JUL-06' AND '23-SEP-07';
```

4. How did the configuration of features composing a region change over a given period?

```sql
SELECT TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP).POLYLINE FROM FEATURE_STATE_TAB p, TABLE(pELEMENTS) L WHERE SDO_INSIDE(P.POLYLINE, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY (1, 1003, 1), SDO_ORDINATE_ARRAY (308009, 274868, 340650, 274868, 340650, 307767, 308009, 307767, 308009, 274868))) = 'TRUE' AND P.VALID_TIME BETWEEN '12-JUL-06' AND '23-SEP-07';
```

5.4.2 Event-based queries

1. What events caused the creation of this feature at time t?

```sql
SELECT L.COLUMN_VALUE.NAME, L.COLUMN_VALUE.EVENT_ID
```
FROM FEATURE_STATE_TAB P, TABLE (P.EVOLVER.CHRONICLE) L WHERE P.TOID = 'osgb4000000021996415' AND P.VALID_TIME = '23-JUL-07';

2. What events caused the features in a region to mutate to their states at time t?

SELECT P.POLYLINE, P.MODIFIER.NAME FROM LINKSTATE_TAB P WHERE P.VALID_TIME BETWEEN '12-JUL-06' AND '23-SEP-07' AND SDO_INSIDE(P.POLYLINE, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARRAY(308009, 274868, 340650, 274868, 340650, 307767, 308009, 307767, 308009, 274868))) = 'TRUE';

3. What events occurred to a feature between times t1 and t2?

SELECT L.COLUMN_VALUE.NAME, L.COLUMN_VALUE.EVENT_ID FROM FEATURE_STATE_TAB P, TABLE (P.EVOLVER.CHRONICLE) L WHERE P.TOID = 'osgb4000000021996415' AND P.VALID_TIME BETWEEN '12-JUL-06' AND '23-SEP-07';

4. What events caused the changes to the features composing a region between times t1 and t2?

SELECT L.COLUMN_VALUE.NAME, L.COLUMN_VALUE.EVENT_ID FROM FEATURE_STATE_TAB P, TABLE (P.EVOLVER.CHRONICLE) L WHERE P.VALID_TIME BETWEEN '12-JUL-06' AND '23-SEP-07' AND SDO_INSIDE(P.POLYLINE, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARRAY(308009, 274868, 340650, 274868, 340650, 307767, 308009, 307767, 308009, 274868))) = 'TRUE';

5. Which features underwent a redirection event between times t1 and t2?

SELECT p.FEATURE_NAME, L.COLUMN_VALUE.TOID, L.COLUMN_VALUE.VALID_TIME FROM FEATURE_STATE_TAB
\[ p, \text{TABLE}(p.\text{ELEMENTS}) \text{ L} \]
\[ \text{WHERE} \ L.\text{COLUMN\_VALUE.\text{MODIFIER.\text{NAME}} = 'Redirection'} \]
\[ \text{AND} \ L.\text{COLUMN\_VALUE.\text{VALID\_TIME BETWEEN '12-JUL-06'}} \]
\[ \text{AND '23-SEP-07'}; \]

5.4.3 Executing queries

Querying the FEM STDBMS can be performed in three ways. One way is to simply execute queries directly using SQL. Another way is to load objects into Java and write methods to perform the searches and manipulations required, as described in chapter 4. Most GISs, however, provide some sort of visualisation in the form of maps, and allow users to interact with these maps and query features using mouse clicks and menus. To evaluate the effectiveness of the FEM system, an interface prototype has been implemented that provides such interaction. This will be described fully as part of the next chapter.

Summary

This chapter has described accessing spatio-temporal data and how the FEM data model facilitates this access. The query search space, and how FEM objects fit into this space, was specified. The FEM model provides navigational access via the graph it creates by establishing relationships between objects as references, and feature histories are accessible via a temporally ordered list of such references. A set of basic spatio-temporal queries have been defined, and examples given of each for ITN data stored in a FEM STDBMS. Queries have been defined from both configuration-based and event-based perspectives. Both of these views of the data are accessible by traversing the graph along their respective routes. The FEM implementation in Oracle Spatial provides temporal and spatial operators, and and the inclusion of events enables the construction of change expressions in queries. The system
can be queried using SQL, Java, or a visual interface, and its performance will be evaluated in the next chapter.
Chapter 6

Evaluation

In section 1.3 it was argued that a temporal GIS could be implemented using standard OO techniques, by producing a conceptual schema to represent complex geographic objects and events, and an implementation to demonstrate visualisation and querying of data. This chapter contains a discussion on how the implementation of the FEM STDBMS has met the objectives set out in section 1.3.2 to prove the hypothesis. Section 6.1 will describe the methods for querying the system and evaluate their effectiveness, and section 6.2 will describe the FEM interface prototype developed to provide visualisation of features and a query interface. Section 6.3 will describe how the prototype executes queries and displays results using point-and-click, and section 6.4 describes how dynamic themes can be formed and displayed using spatio-temporal queries. Section 6.5 will evaluate the querying capabilities of the prototype, and section 6.6 will detail the limitations imposed by the FEM implementation.

6.1 Querying techniques

As mentioned in the previous chapter, the FEM STDBMS can be queried directly using SQL, or in Java by loading FEM objects into Java as custom
objects using JDBC strongly typed interfaces, as described in section 4.2.2. These two methods will now be evaluated.

6.1.1 Querying with SQL

This can be achieved by simply entering SQL queries using an Oracle SQL interface such as iSQL Plus. This provides a text area in which to type the queries and execute them, displaying the results in table format. This is satisfactory only if you are interested in the aspatial attributes of the data, for example ‘How many states are stored for this feature and what are their valid times?’. However, for spatial queries, this is unsatisfactory, as the results will be rows of spatial objects in text form, including large sequences of coordinates, which are of little or no use without a visual reference. Clearly, SQL queries must be at the heart of a temporal GIS, but there must be a visual element to make the results of the queries meaningful.

6.1.2 Querying using Java

Querying in Java has the advantage of preserving the object-oriented nature of FEM objects by minimising the use of relational-style SQL, and enabling these objects to be manipulated efficiently in Java applications. This is particularly useful if a large degree of processing is required, as was the case with the event analysis function described in section 4.4. However, Java querying suffers from the same shortcomings as SQL querying when spatial data is being queried, and is of limited use without a visual component.

6.2 The FEM interface prototype

Because of the limitations of the aforementioned methods of querying, an interface prototype was developed to provide a visual interface with which to
visualise and query features in the FEM STDBMS. This prototype will now be described.

6.2.1 Software and development environment

Oracle Application Server MapViewer [Murray (2006)], henceforth called simply MapViewer, is a programmable environment which can render maps from data in Oracle Spatial. MapViewer has a powerful Java API, and so it is logical to continue the application development using the powerful combination of Oracle Spatial, Java, and MapViewer. Some definitions of important terms used are described as follows:

**Styles** A style is a visual element that is used to represent a spatial feature such as a point, line, or area, and is typically stored in the database with a unique name.

**Themes** A theme is a visual representation of a feature layer. A theme is associated with a table column of type `SDO_GEOMETRY`, and will have a style applied to it to represent the geometry attribute and, optionally, labelling. A theme can be predefined, and stored permanently in the database, or generated dynamically by JDBC queries (known as JDBC themes).

**Maps** Maps are typically a collection of themes. Maps can be composed of a set of predefined themes in the form of a base map, and have dynamic themes added to them in the application. Other elements can also be added, such as a title, map legend, and background image. MapViewer has an associated tool called MapBuilder [Murray (2006)] that can be used to set up themes and base maps.

MapViewer consists of a rendering engine and a set of programming APIs for Extensible Markup Language (XML), Java Server Pages (JSP), and Java. MapViewer connects to an Oracle database using JDBC, and can also load
map metadata from the database, such as map definitions, styles and symbols, and themes that can be applied to data retrieved from queries. The architecture of MapViewer is shown in figure 6.1. An application retrieves maps from the MapViewer server by issuing a map request. A map request has two stages:

- the client requests a map by passing the map name, data source, center location, map size, and, optionally, other data such as dynamic themes

- the server returns the map image and the minimum bounding rectangle (MBR) of the map.

Pre-configured maps can be defined using the MapBuilder [Murray (2006)] tool (described in section 6.2.3). This is a graphical application that can manage the metadata needed to construct a map. A data source relates to a database schema and user, and can be set up using the MapViewer administration page via http.
6.2.2 Requirements of the prototype

The development of the prototype was based on the goal of geovisualisation: to view features, interact with related spatio-temporal data, edit maps by adding and modifying dynamic themes, and to view events and how they impact on features. The following is the list of features that the application must possess in order to fulfill these goals:

- provide visualisation of features as maps
- provide dynamic themes to enable custom visualisations of data as the result of queries without altering the underlying data
- allow interactive querying.

6.2.3 Creating a base map with MapBuilder

Visualising features using the FEM prototype involves creating themes. Although the FEM interface provides a method for generating dynamic themes, a predefined base map makes a good starting point to query features. Because line features are very difficult to select with a mouse pointer, it is useful to include the MBRs of the features as part of the base map to facilitate point-and-click querying.

The MapBuilder tool can be used to set up themes based on the columns of tables containing spatial data, and optionally, to specify labels for features to be displayed. The tool can then combine themes into maps. For the purposes of this evaluation, a base map was set up using MapBuilder from two themes: all polyline attributes of roadlink_typ objects with a state number of 1, and all MBRs from feature_state objects with a state number of 1 with the label of feature_name. Figure 6.2 shows the MapBuilder tool displaying the base map. This constitutes a base map of the base state of the FEM database. The base map is stored in the database and can be retrieved by the FEM prototype when required. Figure 6.3 shows the resultant map as
Figure 6.2: MapBuilder tool used to create a base map
displayed on the viewing area (the canvas) of the FEM prototype. This area displays map images returned from the current map request, and has selectable functions for zooming and panning to help when searching the map. The figure also shows the ‘current themes’ list on the left, which shows all the themes which compose the current map.

The following two sections will describe the way a user can interact with the FEM STDBMS using the interface prototype: by point-and-click, and using dynamic themes. Each query used will be numbered so that it can be referred to the example queries discussed in section 5.3.1.

6.3 Querying features by point-and-click.

This section will describe how features displayed on themes can be queried by using a point-and-click technique. The ‘identify MBR’ radio button shown bottom-left in figure 6.3, makes the highlighted theme (in this case the $MBR\_STATE1$ theme) identifiable. This means that clicking on the map will execute the MapViewer $query\_At\_Point()$ method (query 1), and retrieve data for the MBRs that the mouse pointer touches. The data values to retrieve are specified as parameters of this method. These data values are specified as an array of column names of the table which provides the spatial elements used in the theme. For the prototype, these were set as follows: $feature\_name$, $TOID$, $valid\_time$, and $state\_no$ (column names of the object table $feature\_state\_tab$). The retrieved data is displayed in a dynamic table (see figure 6.4). The query returns results for every $feature\_state$ stored for every feature that is identified. As the base map has feature names as labels, it is easy to match a result from the table to a feature on the map, but some names do not appear at higher resolutions, meaning that panning and zooming of the map is sometimes required to pin-point a feature.

Figure 6.5 shows the result of a point-and-click query after first searching the map for the ‘Penbrey’ feature using the pan and zoom features. It can
Figure 6.3: Base map with MBR theme set as identifiable
Figure 6.4: Result of visually querying a feature
Figure 6.5: Result of visually searching and querying
be seen in the table that the feature details for ‘Penbrey’ have now been identified. This feature will be used to evaluate further some of the querying capabilities of the prototype.

6.3.1 Retrieving configuration and chronicle data

A feature_state’s configuration is retrieved by selecting the configuration data view (shown in figure 6.6) and selecting the appropriate row in the ‘feature detail’ table. This executes an object-relational query to retrieve the data and displays the result in another dynamic table called ‘link detail’. In the following example, the configuration for state 1 for ‘Penbrey’ is retrieved (see figure 6.6). When the appropriate row is selected, query 2, shown below, is executed:

Query 2

```sql
SELECT TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP).TOID, TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP).VALID_TIME, TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP).STATE_NO FROM FEATURE_STATE_TAB p, TABLE(p.ELEMENTS) L WHERE p.TOID = ‘osgb4000000013332209’ AND P.STATE_NO = 1;
```

The above query retrieves the TOID, valid_time, and state_no attributes for each roadlink_typ object referenced in the feature_state’s elements nested table. The TREAT function is required to distinguish the roadlink_typ objects from other types in the inheritance hierarchy, and the DEREF function accesses the referenced objects. The dot operator is used to access specific attributes of the referenced objects. The nested table elements is identified by the TABLE function, and as this table contains only objects and has, therefore, one column, the COLUMN_VALUE function is used to identify
Figure 6.6: Configuration for state 1 of the ‘Penbrey’ feature
it. The variables in the query, the feature_state’s TOID and state_no attributes, are passed to the query dynamically from the selected row of the feature_detail panel.

By simply selecting the chronicle option of the data view, chronicle data will be displayed in the ‘link detail’ table when the appropriate row is selected in the ‘feature detail’ table. This is achieved by retrieving the data referenced by the chronicle attribute of the transition object that is referenced by the selected feature_state. The selection action executes query 3, and figure 6.7 shows the result when state 2 of ‘Penbrey’ is selected:

**Query 3**

```sql
SELECT L.COLUMN_VALUE.name, L.COLUMN_VALUE.EVENT_ID FROM FEATURE_STATE_TAB P, TABLE (P.EVOLVER.CHRONICLE) L WHERE P.TOID = 'osgb400000013332209' AND P.STATE_NO = 2;
```

This query is a lot simpler because there are no subtypes involved, the chronicle nested table of the transition object is accessed using the dot operator on the evolver attribute of the feature_state.

### 6.3.2 Querying an evolved_feature object

A feature’s states can be directly queried if the TOID is known. This is achieved by entering the TOID in the appropriate text box of the ‘Query evolved feature’ panel (see figure 6.8). Clicking the ‘Go’ button executes query 4 for the ‘Penbrey’ feature. This is a very simple query, and involves only a select statement.

**Query 4**

```sql
142
```
Figure 6.7: Event chronicle for state 2 of the ‘Penbre’ feature
SELECT FEATURE_ID, INITIAL_TIME, CURRENT_TIME, STATE_COUNT FROM EVOLVED_FEATURE_TAB WHERE FEATURE_ID = 'osgb4000000013332209';

Figure 6.8 shows the result of this query, displayed in a dynamic table. Selecting the feature details in this dynamic table displays the same information for feature_states as the MapViewer queryAtPoint() method (executed visually), as described in section 6.3.2, but the query is different. Here, the evolution_path attribute of the evolved_feature object is queried, as follows in query 5:

Query 5

EVOLVED_FEATURE_TAB p, TABLE(p.EVOLUTION_PATH) L
WHERE p.FEATURE_ID = 'osgb4000000013332209';

The advantage of this query is that it populates the ‘Feature_detail’ table with data for states for this feature only. Both the configuration and chronicle data views are accessible as previously described.

6.4 Using dynamic themes

So far queries have been executed by interacting with a pre-configured base map by using a point-and-click technique. The FEM prototype allows the base map to be enhanced with the addition of dynamic JDBC themes that display features that are the result of user-defined SQL queries, and examples of these themes will now be provided.

6.4.1 Overlaying features

In the next example, state 2 of the ‘Penbrey’ feature is overlayed onto state 1 with such a theme. By entering the TOID and state number into the appropriate text boxes of the ‘Dynamic Themes’ panel, the prototype can automatically generate the necessary SQL to achieve this, (see figure 6.9). The query is the same as that needed to retrieve a feature_state’s configuration, except that only the spatial attribute (polyline) is required, and is as follows:

Query 6

\[
\begin{align*}
\text{SELECT} & \quad \text{TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP).POLYLINE FROM FEATURE_STATE_TAB p,} \\
& \quad \text{TABLE(p.ELEMENTS) L WHERE p.TOID = 'osgb4000000013332209'} \\
& \quad \text{AND p.STATE_NO = 2;}
\end{align*}
\]
Figure 6.9: Adding a dynamic theme

Figure 6.10 shows the result of this overlay. As you can see, this overlay is not very useful, as the changes between states are at a relatively small scale. Zooming in would make the changes more pronounced, as can be seen in figure 6.11.

The following query enhances a specific area of the base map, represented by the MBR of the B4385 feature, with link objects that have a valid_time attribute value of 03-SEP-06:

**Query 7**

```sql
SELECT P.POLYLINE FROM LINKSTATE_TAB P WHERE P.VALID_TIME = '03-SEP-06' AND SDO_INSIDE (P.POLYLINE, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY (1, 1003, 1), SDO_ORDINATE_ARRAY (241157, 174793, 379572, 174793, 379572, 219140, 241157, 219140, 241157, 174793 ))) = 'TRUE';
```

Figure 6.12 shows the result of the dynamic theme overlayed onto the
Figure 6.10: Overlay of states 1 and 2 of the ‘Penbrey’ feature
Figure 6.11: Figure 6.10 zoomed in to show differences between states
base map. The white sections of the lines show where state 2 links have been overlaid. It can be seen that no state 2 links are visible outside the MBR of the B4385 feature. Figure 6.13 shows a zoomed area of the map that shows sections of the two overlayed states.

6.4.2 Generating events

The FEM interface prototype provides an easy method of enhancing the base map with events. Figure 6.14 shows the ‘Dynamic Themes’ panel where this can be achieved. The user must specify two dates in the boxes labelled $T1$ and $T2$ between which events occurred. Clicking the Add SQL button then generates the SQL query to retrieve the data to form the theme. This query is as follows:

\begin{verbatim}
Query 8
SELECT P.POLYLINE, P.MODIFIER.NAME
FROM LINKSTATE_TAB P
WHERE P.VALID_TIME BETWEEN '12-JUN-06' AND '23-SEP-07'
\end{verbatim}

Figure 6.15 shows the base map enhanced by this dynamic theme. The white areas of the line features represent sections of the features where events have occurred between the specified dates. As in previous examples, it is necessary to zoom in to view details of events in particular areas (see figure 6.16).

6.4.3 Event-based spatio-temporal queries

More specific temporal and spatio-temporal queries can be used in dynamic themes to visualise events. Some examples will now be given to demonstrate this. All the examples involve entering the SQL into the SQL area of the ‘Dynamic themes’ panel of the FEM interface.
Figure 6.12: Base map enhanced with link objects of state number 2 within the area of the B4385
Figure 6.13: Figure 6.12 zoomed in to show differences between overlayed states
Figure 6.14: Dynamic themes panel showing the generation of JDBC theme to display events between two time
Figure 6.15: Base map enhanced with events using a dynamic theme
Figure 6.16: Figure 6.15 zoomed in showing creation events
In the first example, state 2 of the ‘Penbrey’ feature is overlayed onto state 1 and the links for state 2 labelled with the event names. This is achieved as a dynamic theme using the following query:

**Query 9**

```sql
SELECT TREAT(DEREF(L.COLUMN_VALUE) AS ROADLINK_TYP).POLYLINE, L.COLUMN_VALUE.MODIFIER.NAME FROM FEATURE_STATE_TAB p, TABLE(p.ELEMENTS) L WHERE p.TOID = ‘osgb400000013332209’ AND P.STATE_NO = 2;
```

This query overlays both links for state 2 of the ‘Penbrey’ feature with state 1 along with the event names. In the following figures, the nodes have also been added as dynamic themes to show more clearly how the links have changed. To visualise the changes between states, the user must zoom in on the feature. Figure 6.17 shows the right end of the feature, where the node has not moved. Figure 6.18 shows the central area, showing a node has moved its position causing the redirection event for link osgb400000013049080. Figure 6.19 shows the left end, where again the node has changed position, causing the relocation event for link osgb400000013049118.

The next example is a spatio-temporal query that retrieves all events in a specified area between two dates. The query that is required is essentially query 4 described in section 5.4.2 modified to retrieve the relevant link geometries to display on the map. The query is given below:

**Query 10**

```sql
SELECT P.MODIFIER.name, P.POLYLINE FROM LINKSTATE_TAB p WHERE P.VALID_TIME BETWEEN ‘12-JUL-06’ AND ‘23-SEP-07’ AND SDO_INSIDE(P.POLYLINE, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1),
```

155
Figure 6.17: Event overlay of states 1 and 2 of the ‘Penbrey’ feature - right end view
Figure 6.18: Event overlay of states 1 and 2 of the ‘Penbrey’ feature - centre view
Figure 6.19: Event overlay of states 1 and 2 of the ‘Penbrey’ feature - left side view
Figure 6.20: Events that have occurred between two times within a specified area
In this example, the roadlink_types’ polyline geometries and event names from the events referenced by the roadlink_types’ modifier attributes, are retrieved from within the specified spatial region, for roadlink_typ objects whose valid_time values fall between the two specified dates. The spatial region in this case is the MBR for the ‘B4385’ feature. The dynamic theme thus generated and overlayed onto the base map is shown in figure 6.20. As with previous examples, zooming is required to visualise specific events, as shown in figure 6.21.

The next example retrieves all division events that have occurred between the two specified dates:

**Query 11**

```
SELECT L.POLYLINE, L.MODIFIER.NAME FROM LINKSTATE_TAB L WHERE L.modifier.name = 'Division' AND L.VALID_TIME BETWEEN '12-JUL-06' AND '23-SEP-07';
```

This query is much simpler as it does not involve restricting the results to a specific spatial region. All roadlink_typ polyline geometries whose valid_time values fall between the two specified dates are retrieved, along with the event names from the events referenced by the roadlink_typ’s modifier attributes. The base map enhanced with the resultant dynamic theme is shown in figure 6.22. Figure 6.23 shows this view zoomed in to show a division event.

The next example combines the previous ones by retrieving links that have undergone a specific event that has occurred in a specific spatial region between two specified dates:
Figure 6.21: Figure 6.20 zoomed in showing a redirection event
Figure 6.22: Division events that have occurred between two times
Figure 6.23: Figure 6.22 zoomed in to a specific event
Query 12

```
SELECT P.POLYLINE, P.MODIFIER.NAME FROM LINKSTATE_TAB P WHERE P.VALID_TIME BETWEEN '12-JUL-06' AND '23-SEP-07'
AND P.MODIFIER.NAME = 'Redirection' AND SDO_INSIDE (P.POLYLINE, SDO_GEOMETRY (2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARRAY(272070, 195998, 352491, 195998, 352491, 241885, 272070, 241885, 272070, 195998 )))= 'TRUE';
```

This query retrieves the roadlink_typ polyline geometries and event names from the events referenced by the roadlink_typs’ modifier attributes that correspond to ‘Redirection’, from within the specified spatial region, whose roadlink_typ objects valid_time values fall between the two specified dates. In this case the spatial region is the MBR for the ‘A40’ feature. Figure 6.24 shows the resultant themes displayed on the base map, and figure generated and figure 6.25 shows the map zoomed in to show two redirection events.

6.5 Query evaluation

This section will relate the queries executed using the prototype to the example queries outlined in sections 5.4.1 and 5.4.2.

6.5.1 Configuration-based queries

What is the configuration of a feature at time t? This was achieved with queries 2, 5, and 6.

How did a feature’s configuration change over a given period? This was achieved by comparing results from query 2 for different feature_states, and overlaying states as shown in figure 6.10 using query 6.
Figure 6.24: Redirection events between two times within a specified spatial region
Figure 6.25: Figure 6.24 zoomed in to show redirection events
What configuration did the features composing a region have at time $t$? This was achieved with query 1 and query 7.

How did the configuration of features composing a region change over a given period? This can be achieved by modifying query 7 by substituting the ‘$=$’ temporal operator with the between temporal operator and two dates. The event-based query 10, however, provides more descriptive results.

### 6.5.2 Event-based queries

What events caused the creation of this feature at time $t$? This was achieved by query 3 and query 9.

What events caused the features in a region to mutate to their states at time $t$? This can be achieved by query 10 using the ‘$=$’ operator and a single date in place of the between operator.

What events occurred to a feature between times $t_1$ and $t_2$? This can be achieved by query 9 using the between temporal operator in place of the state_no predicate.

What events caused the changes to the features composing a region between times $t_1$ and $t_2$? This was achieved by query 10. Query 8 provides events for the entire base map between two times.

Which features underwent a redirection event between times $t_1$ and $t_2$?

This was achieved by query 12.
It can be seen from this evaluation that the FEM interface prototype performs temporal queries effectively with the point-and-click querying feature. These queries retrieve time relevant attributes only, displayed in the dynamic tables. The prototype also demonstrates that spatio-temporal and spatio-temporal range queries can be executed and displayed using dynamic themes, and that change expressions can be included to enhance the descriptiveness of results. The dynamic themes feature provides a powerful visualisation mechanism based on dynamic spatio-temporal queries.

6.5.3 Prototype evaluation

In chapter 1 a set of criteria for a temporal GIS from Langran (1992) were described. The prototype will now be evaluated in the light of these criteria.

- **Inventory** - the system stores a complete history of stored objects - achieved.

- **Analysis** - the system provides retrieval and visualisation of configurations and events, and spatial analysis is possible using the inbuilt spatial functions of Oracle Spatial - partially achieved.

- **Scheduling** - the system does not store scheduling information - not achieved.

- **Display** - the system provides an interactive visual display - achieved.

- **Updates** - the system enables non-destructive updates to the FEM STDBMS - achieved.
6.6 Limitations of the FEM system

Retrospective updates
This is outside the design parameters of the FEM model. As a compromise was made to include only valid time, the system does not handle retrospective updates. The update algorithms used assume incremental updates, so data from, say, three years ago would be inserted as a new update and have a higher state number than newer data.

Data consistency
There are errors present in the ITN data and the correction of these errors was a non-trivial process. Therefore, the development of the FEM STDBMS and the FEM interface prototype did not attempt to correct these errors. These errors consist mainly of configuration issues in the standard ITN join tables, which did not always maintain referential integrity.

Performance
Although the OO structure of the FEM system would be more complex and difficult to represent in a relational system, it is possible that the implementation of objects in Oracle means that the performance of a FEM STDBMS may not match relational tables. This is because although Oracle provides an OR framework, it ultimately stores nested table data in relational tables and does so by using hidden columns. This incurs a performance overhead, and means that DMLs could run significantly slower and scale less well [Menon (2005)].

Summary
This chapter has described the query methods for accessing spatio-temporal data in the FEM STDBMS, and concluded that the most effective way to do
this was with a visual interface. The FEM interface prototype was described and the querying and display of data was detailed and evaluated against the example queries set out in the previous chapter. It can be seen that the prototype copes well with simple temporal and spatio-temporal queries, and dynamic themes with the more complex temporal and spatio-temporal range queries using dynamic themes.

The visual component of the prototype is enhanced by the dynamic querying capabilities, and visualising different states is clear, with automated SQL generation for single states. Overlays of different states are achieved by creating dynamic themes, but changes are not clear at high resolutions. However, these are more evident at lower resolutions and changes can be labelled with event names adding clarity to the overlay.

The prototype addresses one of the two remaining criteria for an optimal data model outlined in section 3.1.1: integrity. Data integrity is maintained by the maintenance of feature configurations, although this was compromised to a small degree by errors in the ITN data used. The final criteria, the elimination of data redundancy, is not achieved by the implementation. This is because although the system only stores new versions of links and nodes, and maintains feature configurations without duplicating versions, new versions of links’ geometries may contain only minor changes and retain much of the old version’s coordinate data.
Chapter 7

Conclusions

Section 1.3 listed four objectives that if achieved would prove the hypothesis of this research. It is contended that this hypothesis has been proved, and to substantiate this claim, section 7.1 will review each objective against the previously described work. Section 7.2 will summarise the contributions of this research, and section 7.3 will describe possible future work.

7.1 Proof of hypothesis

To use object-oriented design techniques to develop a conceptual data model for the representation of spatio-temporal geospatial data.

- It was argued in chapter 2 that previous attempts at a spatio-temporal representation fell short of the requirements of a temporal GIS. OO approaches were more successful, but lacked support for all of the required features, particularly complex objects. Chapter 3 introduced the FEM conceptual data model which utilises the best aspects of the previous OO approaches, but includes object versioning techniques established in the CAD/engineering design domain and represents events.
• Chapter 2 argued that a key problem when designing a temporal GIS was the increased complexity introduced by the inclusion of time. The FEM design eliminated much of this complexity by restricting this to valid time only. It is argued that the resultant model is suitably elegant.

• Chapter 3 emphasised the OO nature of the FEM conceptual model, which utilises the OO features of inheritance, encapsulation, references, and collections to construct the data types that represent its objects and their relationships.

• Section 3.1.1 detailed the criteria necessary for an optimal data model. The FEM conceptual data model fulfills all of these criteria except the elimination of redundancy.

To develop queries for retrieving and analysing spatio-temporal geospatial data.

• Chapter 5 described the process of accessing data in the FEM implementation, and included a set of example spatio-temporal queries that should be possible.

• Section 5.1 described how the FEM system enables queries to be performed from configuration-based and event-based views.

• Such queries were specified, executed, and evaluated in chapter 6.

• Spatial analysis is possible through the available functions in Oracle Spatial.

• Chapter 6 described dynamic queries that define themes that can be used for visual analysis of change between different states of features.
To implement an object-oriented temporal GIS based on the conceptual data model to enable visualisation and querying of spatio-temporal geographic data.

- Chapter 4 described the implementation of a STDBMS based solely on the FEM conceptual data model.

- Chapter 6 described the FEM interface prototype developed to enable the querying and visualisation of features in the FEM STDBMS, and evaluated its effectiveness as a temporal GIS. The evaluation of the prototype established that the system fulfills three from five of Langran’s criteria for a temporal GIS fully, and a further single criterion partially.

To test and evaluate the data model and the implemented system.

- The FEM conceptual data model was designed to represent complex geographic data. Section 3.3.1 established that the data used to implement the FEM system, ITN data, is representative of such data.

- Section 6.3.7 evaluated the querying capabilities of the FEM prototype, and concluded that simple temporal queries could be performed effectively using a point-and-click technique, while temporal and spatio-temporal range queries could be performed using dynamic themes.

- Chapter 6 established the visual capabilities of the prototype to be effective.

- Section 6.4 identified the areas of retrospective updates and data consistency as limitations of the system.

It is argued that having proved that the objectives supporting the hypothesis of this thesis have been achieved, that the hypothesis, therefore, has been proved.
7.2 Contributions of this research

The contributions of this research are as follows:

1. An OO spatio-temporal conceptual data model which can be utilised by any OO or OR SDBMS and OO programming language. The model is elegant yet sufficiently expressive to represent complex geographic objects and events within a limited number of types. The model provides extensibility by using the OO feature of inheritance, and should be applicable to other representative datasets such as those described in section 3.3.1. The model incorporates the concept of events as objects in space and time, and represents spatial and temporal relationships between objects and events.

2. The utilisation of object versioning techniques established in the fields of CAD and engineering design within an absolute-hybrid spatio-temporal model. No other model, to our knowledge, uses these techniques to the same extent to represent spatio-temporal data.

3. An implementation of the model using ITN data in the form of a temporal GIS. The GIS fulfills three out of five of Langran’s requirements for a temporal GIS fully, and one partially.

7.3 Future work

There are a number of directions that future work in the area of this research could take, and these are now described.

Improved data consistency and performance

From a practical point of view, the most logical course would be to rectify some the shortcomings of the implementation described in section 6.4, specifically data consistency and performance. Improving data consistency
would entail detecting and correcting errors in the ITN data before converting them to FEM objects. The performance issues arise as a result of Oracle’s implementation of its OR features, and so cannot be overcome using Oracle Spatial. Implementing FEM in another SDBMS, therefore, could be investigated. It would also be useful to implement a FEM system using an alternative dataset, such as one of those described in section 3.3.1.

Extension of the model

One way in which the FEM model could be extended would be to include more attribute information. This data is supplied in the standard ITN data and is called road routing information. The data includes information such as speed limits, road direction (one way roads), and the positions of traffic lights and signs. The inclusion of this data would enable comparative analysis between different states of road features, for example in accident analysis. There is also potential to model network flows and assess the impact of interruptions to flows by road works and accidents. Work has been carried out in this area by Galton and Worboys (2005). The inclusion of this functionality would constitute full-blown analysis, and also enable scheduling of road alterations and maintenance.

Development of the prototype

The interface prototype as it stands represents a very basic temporal GIS in terms of its functionality. The application could be extended to accommodate any new analysis functions that might be developed. It could also be extended to automatically import datasets, and provide more automation in the way it displays features. Providing more automated generation of SQL would also be advantageous.
**Apply the model to multiple representation**

The prospect of applying the FEM model to the application of multiple representations of geographic features in the domain of vernacular geography could be investigated. Vernacular geography [Twaroch et al. (2008)] encompasses areas that are often expressed by users in terms not usually found in geographic databases, such as ‘down town’. Such features are also often required at varying levels of detail, and with different emphasis, for example all the music shops in an area.

URL: http://www.oracle.com/pls/db10g/


URL: http://giswww1.bcit.ca/jcandy/thesis/


ESRI (2005), ‘Arcsde overview’.


ESRI (2008), ‘An overview of the geodatabase’.

URL: webhelp.esri.com/arcgisdesktop/9.2/index.cfm


180


**URL**: www.oracle.com/technology/products/database/

Lorentz, D. and Gregoire, J. (2003), *Oracle Database SQL Reference 10g Release 1 (10.1)*.

**URL**: http://www.oracle.com/pls/db10g/ Accessed 1st September 2008


**URL**: citeseer.ist.psu.edu/medeiros94using.html Accessed 1st September 2008


URL: citeseer.ist.psu.edu/612397.html Accessed 1st September 2008


URL: citeseer.ist.psu.edu/gisconsortium97opengis.html Accessed 1st September 2008


URL: http://www.ordnancesurvey.co.uk/oswebsite/products/osmastermap/userguides/index.html


Peuquet, D. J. (2003), Representations of space and time, New York, Guilford.


183
URL: citeseer.ist.psu.edu/697114.html Accessed 1st September 2008


URL: citeseer.ist.psu.edu/raza96design.html Accessed 1st September 2008


**URL:** citeseer.ist.psu.edu/sotiropoulou98temporal.html Accessed 1st September 2008


**URL:** http://www.census.gov/geo/www/tiger/ Accessed 1st September 2008


URL: http://www.geog.port.ac.uk/hist-bound/ Accessed 1st September 2008


186

